2017

Comparative Research on Robot Path Planning Based on GA-ACA and ACA-GA

Chenhan Wang
University of Windsor

Comparative Research on Robot Path Planning Based on GA-ACA and ACA-GA

By

Chenhan Wang

A Thesis
Submitted to the Faculty of Graduate Studies
through the School of Computer Science
in Partial Fulfillment of the Requirements for
the Degree of Master of Science
at the University of Windsor

Windsor, Ontario, Canada

2017

Comparative Research on Robot Path Planning Based on GA-ACA and ACA-GA

by

Chenhan Wang

APPROVED BY:

E. Abdel-Raheem
Department of Electrical and Computer Engineering

J. Chen
School of Computer Science

D. Wu, Advisor
School of Computer Science

December 15, 2017

DECLARATION OF ORIGINALITY

I hereby certify that I am the sole author of this thesis and that no part of this thesis has been published or submitted for publication.

I certify that, to the best of my knowledge, my thesis does not infringe upon anyones copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in my thesis, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. Furthermore, to the extent that I have included copyrighted material that surpasses the bounds of fair dealing within the meaning of the Canada Copyright Act, I certify that I have obtained a written permission from the copyright owner(s) to include such material(s) in my thesis and have included copies of such copyright clearances to my appendix.

I declare that this is a true copy of my thesis, including any final revisions, as approved by my thesis committee and the Graduate Studies office, and that this thesis has not been submitted for a higher degree to any other University or Institution.

ABSTRACT

The path planning for mobile robots is one of the core contents in the field of robotics research with complex, restrictive and nonlinear characteristics. It consists of automatically determining a path from an initial position of the robot to its final position. Due to classic approaches have several drawbacks, evolutionary methods such as Ant Colony Optimization Algorithm (ACA) and Genetic Algorithm (GA) are employed to solve the path planning efficiently.

Firstly, grid method is used to establish the environment model, and some modifications are made to accommodate ACA to path planning in a grid-based environment. Besides, genetic operators were introduced to the fundamental ACA (GA-ACA, ACA-GA), using the crossover and mutation operators to expand the search space and enhance the overall solution in the previous research work.

This thesis mainly introduces these two hybrid algorithms, GA-ACA and ACA-GA, and will compare the performance of them under multiple grid maps in static environments.

To verify the effectiveness of these two hybrid algorithms, a path planning simulation system for mobile robots is designed based on MATLAB development environment. The experiment results show that the algorithm efficiency of GA-ACA and ACA-GA is better than that of the traditional GA and ACA algorithms, and it is more suitable to apply ACA-GA than GA-AGA regarding algorithms' convergence speed and stability in a complicated environment map.

AKNOWLEDGEMENTS

I would like to express my sincere appreciation to my supervisor Dr. Dan Wu for his constant guidance and encouragement during my whole Master's period in the University of Windsor. Without his valuable help, this thesis would not have been possible.

I would also like to express my appreciation to my thesis committee members Dr. Esam Abdel-Raheem, and Dr.Jessica Chen. Thank you all for your valuable guidance and suggestions to this thesis.

Last but not least, I want to express my gratitude to my parents and my friends who give me consistent help over the past three years.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

Introduction

1.1 Research Background

With the development and maturation of computer technology, control theory, aritifical intelligence theory and sensor technology, the research of robot has entered an entirely new phase. Mobile robots, as an important branch of robotics, has received wide recognition in an academic area all over the world.

A mobile robot is an automatic machine which has the capability to move around in the environment [28]. Mobile robots can be "autonomous" which means they are capable of navigating an uncontrolled environment without the need for physical or electro-mechanical guidance devices.

From 1966-1972, the Stanford Research Institute was building and doing research on Shakey the Robot [25], which was the first robot that can avoid obstacles automatically. In 1999, Sony introduced AIBO [31], a robotic dog capable of seeing, walking and interacting with its environment. In 2003, QRIO [16] was created by Sony to follow up on the success of its AIBO entertainment robot. QRIO is the first humanoid robot that can accomplish many actions such as running, jumping, throwing.

In recent years, mobile robots have become more commonplace in industrial, agricultural and commercial settings. In all applications of mobile robots, they perform the navigation tasks using the following building blocks [26] in Fig.1. We can see that navigation of a mobile robot involves perception of the environment, localization and map building, cognition and path planning and motion control in Fig.1. Among the four steps of mobile robot navigation, path planning is an essential one. The main

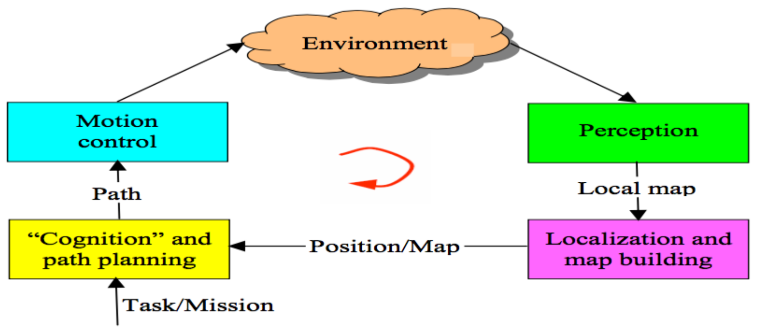


FIGURE 1: Building Blocks of Mobile Robot Navigation.

objective of path planning is about how to make mobile robots choose a collision-free path according to the environment information.

1.2 Mobile Robots Path Planning

Path planning of a mobile robot is to determine a collision-free path from a start point to a goal point optimizing a performance criterion such as distance, time or energy, distance being the most commonly adopted criterion [26]. Based on the availability of information about environment, there are two categories of path planning algorithms, namely off-line and on-line. Off-line path planning of robots in environments where complete information about stationary obstacles and trajectory of moving obstacles are known in advance is also known as global path planning. When complete information about environment is not available in advance, mobile robot gets information through sensors, as it moves through the environment. This is known as on-line or local path planning. Both off-line and on-line path planning algorithms can be categorized into classic approaches and evolutionary approaches.

1.2.1 Off-line Path Planning Algorithms

Examples of path planning in off-line environments are service robots operating during maintenance period of a nuclear power plant, automated guided vehicles in a factory, where there are not any changes in the environment map [26]. The off-line path planning algorithms can be categorized into classic approaches and evolutionary approaches.

Classic Approaches

1. **Configuration space approach [20]**:

   The central idea of configuration space approach is the representation of the robot as a single point. Thus, the mobile robot path planning problem is reduced to a 2-dimensional problem. As robot is reduced to a point, each obstacle is enlarged by the size of the robot to compensate. Then, we construct a configuration space to describe the robot and its surroundings by using some basic shapes such as a predefined convex polygon. The configuration space is represented as a connected graph, and then the path planning is performed by searching the connected graph. This method is flexible, but the complexity of this algorithm is proportional to the number of obstacles, and it can not guarantee that the method can output the shortest path under any circumstances. That is one of the drawbacks of most classic approaches.

2. **Visibility graph approach [20]**:

   The Visibility graph approach is drawn by joining two vertices of mutually visible polygonal obstacles that are present between start and target points. The word "Visibility" refers to the requirements of the robot between the vertices of obstacles, the target point between the vertices of obstacles and the vertices of one obstacle between the vertices of another obstacle can not cross the obstacles. The shortest path is then identified through the roads obtained from the visibility graph. Using some optimization algorithms can remove some unnecessary connecting lines to simplify the visibility graph, thus shortening the search time. However, these optimization algorithms sometimes lacks flexibility, and once the starting point and target point change, the visibility graph must be reconstructed leading the low search efficiency.

3. **The cell decomposition approach [19]**:

The cell decomposition approach computes the configuration space of the mobile robot decomposes the resulting space into cells and then searches for a route in the free space cell graph. Among the cell decomposition approaches, Grid method [2] is the most popular one where grids are used to generate the map of the environment. The main difficulty is how to find the size of the grids, the lesser the size of grids, the more accurate will be the representation of the environment. However, using lesser grids will result in exponential rise in memory space and search range [14]. In my thesis, I will use the grid method to establish robot motion environmental maps in chapter 3.

Evolutionary Approaches

Classic approaches sometimes take more time when selecting a feasible collision-free path. Also, classic approaches tend to get locked in local optimal solution which may be far inferior to the global optimal solutions. Moreover, path planning of a mobile robot in the presence of multiple obstacles is found to be a non-deterministic polynomial time hard (NP-hard) problem[3]. It becomes even more complicated when the environment is dynamic. These drawbacks make the classic approaches incompetent in complex environments. Hence, evolutionary approaches such as Genetic Algorithm (GA), Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO) and Simulated Annealing (SA) is employed to solve the path planning problem efficiently.

1. **Genetic Algorithm (GA)**:
   GA is an optimization tool based on the mechanics of natural genetics and selection. The first step in path planning using GA is a random generation of the population containing alternative paths. [9] presented a visibility-based repair approach that is used to quickly transform invalid paths into valid paths and then subject to binary coded GA. GA with binary string is computationally costly for the reason that before each evaluation of function, chromosomes are transformed to phenotypes. [33] presented a genetic based path planning algorithm, in which populations are generated including invalid paths. Later such invalid path sequences are subjected to penalty function evaluation. This

increases computation load resulting in higher execution time. The specific process of how does GA works will be introduced in Chapter 4.

2. **Particle Swarm Optimization (PSO)** [17]:
   Particle Swarm Optimization (PSO) is a widely used evolutionary algorithm in path planning. It is an evolutionary computation technique inspired by social behavior of bird flocking or fish schooling. Years of study on the dynamics of bird and fish resulted in the possibilities of utilizing this behavior as an optimization tool. Compared to GA, the advantages of PSO are that PSO is easier to implement and there are fewer parameters to be adjusted.

3. **Ant Colony Optimization (ACO)** [4]:
   ACO is inspired by the foraging behavior of ants for finding the shortest path to the food source. The specific content about ACO and mobile robot path planning based on ACO algorithm is introduced in Chapter 2 and 3.

4. **Simulated Annealing (SA)**:
   Simulated Annealing (SA) is a type of heuristic random search method and it resembles the cooling process of molten metals through annealing. [21] present a method employed SA for collision-free path amid static polygonal obstacles in configuration space setting. [23] developed SA algorithm based approach which used vertices of the static and dynamic obstacles as search space for dynamic environments.

1.2.2 On-line Path Planning Algorithms

Applications of path planning in on-line environments include planet exploration, mine industry, reconnaissance robots, etc [26]. Nowadays, evolutionary approaches are increasingly being used along with classic approaches.

Classic Approaches

1. **Artificial Potential Field (APF) approach [18]** :

   The main idea of APF approach is that a point robot moves under the influence of an APF in which obstacles are assumed to generate repulsive forces and the target is assumed to generate attractive forces. The robot moves as per the resultant of these forces. This approach is known for its mathematical elegance and simplicity as path is found with very little computation. However, the drawback of this approach is that robot may become stagnant or trapped when there is a cancellation of equal magnitudes of attractive and repulsive forces.

2. **Vector Field Histogram approach [1]**:

   The central idea of this approach is that a polar histogram is generated to represent the polar density of obstacles around a robot at every instant. The robots steering direction is chosen based on the least polar density and closeness to the goal. In a given environment, the polar histogram must be regularly regenerated for every instant and hence the method is suited for environments with sparse moving obstacles.

3. **Velocity obstacle approach [11]**:

   This method consists of choosing avoidance maneuvers to avoid static and moving obstacles in the velocity space. They used basic heuristic strategy for prioritizing objectives such as averting collisions, attaining the goal or accomplishing trajectories with preferred topologies.

4. **Dynamic windows approach [12]**:

   The dynamic window approach contains the feasible linear and angular velocities taking into consideration acceleration capability of robot. Then the velocity at the next instant is optimized for obstacle avoidance, subject to vehicle dynamics.

Evolutionary Approaches

When applying on-line path planning algorithms, computation time is the most important aspect that we should consider. However, with classic approaches, the results can hardly be achieved in very quick time because of incomplete information of the environments.Therefore, classic approaches are often combined with evolutionary approaches like GA, PSO, etc. to overcome their drawbacks.

1. **Evolutionary APF [30]**:
   The evolutionary APF algorithm is to derive optimal potential field functions using GA. When the robot is trapped, a separate algorithm named escape-force is introduced to recover from a trap.

2. **APF combined with SA [22]**:
   This approach considers the problems of goal non-reachable with obstacles nearby (GNRON) and local minima in soccer robots. New potential functions have been derived by considering the distance information of start and target points for GNRON problem.

1.3 Thesis Motivation and Statement

The thesis is mainly focused on robot path planning problem based on ant colony optimization algorithm and two ant colony algorithms combined with genetic algorithms, namely GA-ACA and ACA-GA. The previous research work of [15] and [35] indicated that GA-ACA and ACA-GA perform better than the traditional algorithm, ACA. However, the researchers had not compared the performance of GA-ACA and ACA-GA in the previous work. This motivates me to do the further study on these two algorithms.

Chapter 2 introduces the concept of ant colony optimization algorithm (ACA) and describes how it is inspired by the nature phenomenon. Then, we give detailed explanation of ACA in theory and describe three main ACA algorithms: Ant System, MAX-MIN Ant System, Ant Colony System. Last, we explore the research directions related to ACA and how it applies to the NP-hard problems.

Chapter 3 establishes the robot motion environmental model by using grid method and introduces two main methods for grid making. Then, we describe how ant colony optimization algorithm (ACA) for robot path planning works under grid environmental maps.

In chapter 4, two algorithms based on the ant colony algorithm are proposed: GA-ACA algorithm and ACA-GA algorithm. We describe the process of GA-ACA and ACA-GA respectively and compare the differences between these two hybrid algorithms.

In chapter 5, based on MATLAB platform, we design four comparative experiments to verify the validity and effectiveness of the GA-ACA and ACA-GA algorithm under different grid maps.

The chapter 6 summarizes the work of this thesis and points out the limitations of this research, and looks forward to the following works to do for this research study.

CHAPTER 2

Review on Ant Colony Optimization

2.1 Introduction to Ant Colony Optimization

Ant Colony Optimization (ACO) was first introduced and proposed by Marco Dorigo in his Ph.D. thesis in 1992 [4]. It is a metaheuristic that inspired by the food seeking behavior of ant colony. As we all know, in the natural world, the ant is a kind of social insect. Although the behavior of a single ant looks quite simple, multiple ants can cooperate to construct a huge social group to accomplish much more complicated tasks. When the ants search for food, initially they just wander randomly to explore the area near their nest, and different ants can choose different paths to explore due to their random behavior manner. As soon as one of the ants successfully locate the food source, this ant will carry bearable amount of food back to its nest. During their return journey, assuming that the ants remember its traveled route from the nest to the located food source, it will return to the nest along the same route, and leave chemical pheromone trail on the ground. The amount of the pheromone left may vary depending on the quality and quantity of the food. Then, other ants now can choose the route that has denser pheromone (which can be treated as a better route) and guides them to the food source. This behavior will also cause the originally better route becomes even much better by aggregating more pheromones.

This ant colony behavior can be modeled as metaheuristic, which is the core concept of Ant Colony Optimization algorithm. It belongs to the category of approximate

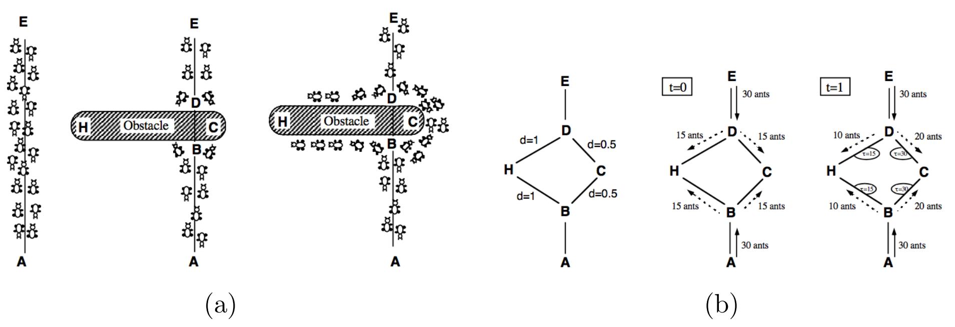


FIGURE 2: Path Choosing based on Pheromone. (a) An example with real ants. (b) An example with artificial ants.

algorithms that are used to obtain good enough solutions to solve hard combinatorial optimization problems (CO). The goal of CO problem is to find out a good enough solution in a reasonable amount of time.

Considering the scenario introduced in [8] as shown in Fig.2 (a), there is a path between food source E and nest A. When an obstacle is placed to cut off the path, ants have to decide whether to turn left or right (position H or C). Firstly there are no pheromones left at path BH, BC, DH and DC , which can be clearly seen in Fig.2 (b); thus the probability of choosing left or right is equal. However, since BCD is shorter than BHD, the ants turned to C will arrive E faster than the ants which turned to H. When the first ant returned from E arrives D, the path DC have more pheromones since more ants have passed D from C than from H, so that this first ant that has finished the trip will have higher probability of choosing path DCB. The consequence is that the pheromone on the shorter path will grow faster than on the longer path, and therefore the probability of choosing paths is quickly biased towards the shorter one.

Having the conclusion we draw in the above paragraph, now considering the abstract model shown in Fig.2(b), 30 ants depart in 1 time slot, the length of BH and HD are both 1, and the length of BC and CD are both 0.5, if we neglect the length of AB and DE, the length of ABHDE (path L) is 2 times larger than ABCDE (path R), then at $t = 1$, assume x ants went back to A from path L, then $2x$ ants should be back to A from path R. Therefore the pheromone density $\tau_R = 2\tau_L$. Thus, for the ants depart at $t = 1$, the probability of choosing path R is $\frac{2}{3}$.

2.2 Theroretical Explanation of ACO

Ant Colony Optimization (ACO) algorithms can be treated as stochastic search procedures. Although multiple ACO algorithms exist, the central component of these algorithms is the pheromone model that is used to sample the search space[6] probabilistically. The model of Combinatorial Optimization (CO) problem can be used to implement the pheromone model. In the following subsection, we will first introduce the model of solving CO problem.

2.2.1 Combinatorial Optimization Problem

[6], [5] mentioned that a CO problem could be defined as: $\mathcal{P} = (\mathcal{S}, \Omega, f)$, where

- $\mathcal{S}$ is a finite solution set; it is defined over a finite set of discrete decision variables X_i, $(i = 1, 2, ..., n)$;
- Ω represents a set of constraints among the discrete decision variables X_i;
- f is an objective function that assigns a cost value to every solution. f can be defined as $f : \mathcal{S} \to \mathbb{R}^+$.

For the solution set $\mathcal{S}$, which can also be called search space, the discrete decision variable X_i are assigned with domain values v_i^j, where $v_i^j \in D_i = \{v_i^1, v_i^2, ..., v_i^{|D_i|}\}$. $s \in \mathcal{S}$ is called a feasible solution when s is a complete assignment that each discrete decision variables X_i has a domain value assigned that satisfies the constriant set Ω. Moreover, a feasible solution $s \in \mathcal{S}$ is called a globally optimal solution only if for all $s^* \in \mathcal{S}$, $f(s^*) \leq f(s)$. Sometimes there is not only one globally optimal solution, the set of globally optimal solutions can be represented as $\mathcal{S}^* \subseteq \mathcal{S}$.

This CO problem model is a finite set of *solution components* and a *pheromone model*. The discrete decision variable X_i and one of its domain values v_i^j is a *solution component* denoted by c_i^j , which can be described as $c_i^j : X_i = v_i^j$. For the *pheromone*

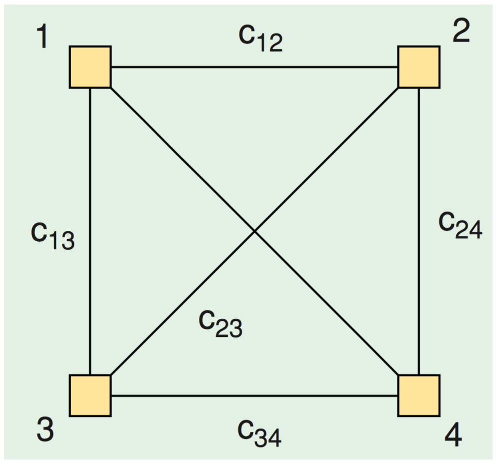
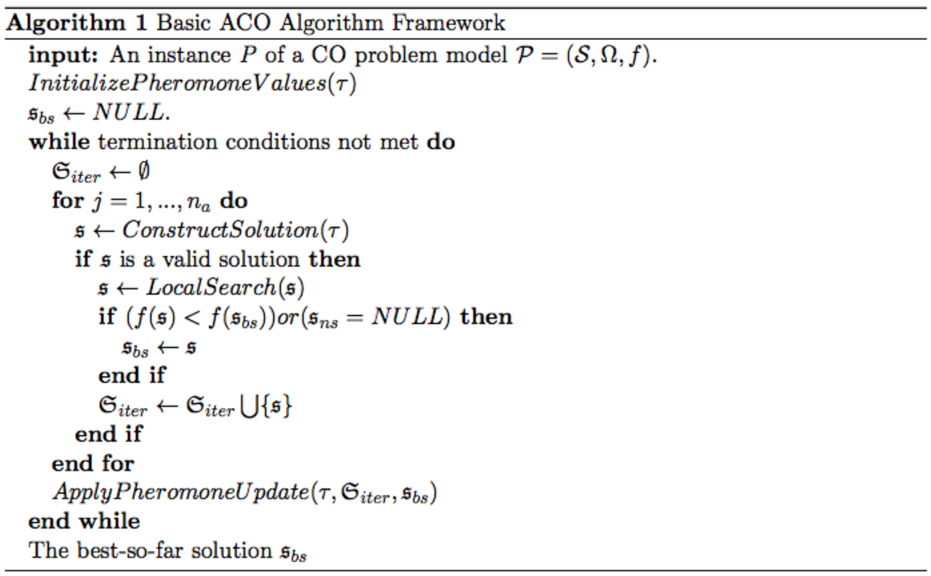


FIGURE 3: Example of Construction Graph in ACO.

model, each of the solution component c_i^j has a pheromone trail parameter τ_i^j. All pheromone trail parameters are denoted by τ. We use $\mathfrak{C}$ to represent all the solution components, which means $\mathfrak{C} = \{c_1^j, c_2^j, ..., c_n^j\}$.

The following is a specific example of the described CO problem. As shown in Fig.3, in ACO problems, an artificial ant builds a solution by traversing the fully connected construction graph denoted by $\mathcal{G}_C = (V, E)$, V is the set of vertices, E is the set of edges. In our simple example, the construction graph contains four vertices, artificial ants move from vertex to vertex along the connected edges. For the solution components $\mathfrak{C}$ we just mentioned, in this case, there are four solution components: c_i^j, $(i = 1, 2, 3, 4)$, which can map to the four discrete variables X_i, $(i = 1, 2, 3, 4)$, and also can represent the four vertices of the construction graph. For example, a solution component c_2^j can represent node 2 in the construction graph, and X_2 indicates the node to be visited after node 2, so $X_2 = c_2^j$ means the next node visited is node j, the value range of j is $|D_i|$, which can be treated as the connected vertices with i. And in general, c_i^j indicates a solution that node j should be visited immediately after node i, in order words, it indicates edge (i, j), ants deposit pheromone τ_i^j on the edges (i, j) describes its importance of reaching the best- optimized solution. The way how an artificial ant builds a solution by traversing the fully connected construction graph will be discussed theoretically in the next section.



```
Algorithm 1 Basic ACO Algorithm Framework
  input: An instance $P$ of a CO problem model $\mathcal{P} = (\mathcal{S}, \Omega, f)$.
  $InitializePheromoneValues(\tau)$
  $\mathfrak{s}_{bs} \leftarrow NULL.$
  while termination conditions not met do
    $\mathfrak{S}_{iter} \leftarrow \emptyset$
    for $j = 1, ..., n_a$ do
      $\mathfrak{s} \leftarrow ConstructSolution(\tau)$
      if $\mathfrak{s}$ is a valid solution then
        $\mathfrak{s} \leftarrow LocalSearch(\mathfrak{s})$
        if $(f(\mathfrak{s}) < f(\mathfrak{s}_{bs})) or (\mathfrak{s}_{ns} = NULL)$ then
          $\mathfrak{s}_{bs} \leftarrow \mathfrak{s}$
        end if
        $\mathfrak{S}_{iter} \leftarrow \mathfrak{S}_{iter} \bigcup \{\mathfrak{s}\}$
      end if
    end for
    $ApplyPheromoneUpdate(\tau, \mathfrak{S}_{iter}, \mathfrak{s}_{bs})$
  end while
  The best-so-far solution $\mathfrak{s}_{bs}$
```

FIGURE 4: Basis ACO Algorithm Framework.

2.2.2 Basis of ACO

Fig.4 shows the framework of basic ACO algorithm [6]. Initially, all pheromone values τ_i^j that belongs to the set of τ (the pheromone model) are initialized to a constant value $c > 0$, this process is finished by the function *InitializePheromoneValues* shown in Algorithm 1. We assume there are n_a artificial ants in total. During each iteration, n_a ants use the current pheromone model to probabilistically construct solutions to the CO problem. For each ant, the function *ConstructSolution* is executed to construct a solution component. This function is also the basic building block component of any ACO algorithm as it is a constructive heuristic for probabilistically constructing solutions. The solution constructing process can be reflected as selecting sequences of elements from the finite set of solution components $\mathfrak{C}$, since the solution is constructed by each ant in each iteration, thus we call this as partial solution, using $\mathfrak{s}^p$ to represent it. The construction process is described as follows:

- Start with the empty partial solution set $\mathfrak{s}^p = \emptyset$;

- At each solution construction step, extend the partial solution set sp by adding a feasible solution component c_i^j from $\mathfrak{C} \setminus \{\mathfrak{s}^p\}$, which means the solution component is in the set $\mathfrak{C}$ but not yet included in the set $\mathfrak{s}^p$.

This process can be visualized as a continuous walk along the edges of the construction graph $\mathcal{G}_C = (V, E)$, all the vertices can be treated as the solution components in the set $\mathfrak{C}$ as we described in the previous section. In each step of the solution construction, the c_i^j that belongs to $\mathfrak{s}^p$ is chosen probabilistically based on the current pheromone model. The probability of choosing a specific solution component c_i^j is proportional to $[\tau_i^j]^\alpha \cdot [\eta(c_i^j)]^\beta$[6]. η is a function that assigns to each feasible solution component, which can be treated as a heuristic value or heuristic information. Parameters α and β are all positive numbers that determine the relative importance of pheromone value and heuristic information. [6] also mentioned the heuristic information is optional, but if included, the ACO algorithm can usually achieve better performance. If considering the heuristic information, the probability of choosing the next solution component c_i^j is defined as follows:

$$P(c_i^j|\mathfrak{s}^p) = \frac{[\tau_i^j]^\alpha \cdot [\eta(c_i^j)]^\beta}{\sum_{c_i^k \in \mathfrak{s}^p} [\tau_i^k]^\alpha \cdot [\eta(c_i^k)]^\beta} \tag{1}$$

This equation is also called *transition probabilities*, which is first used in [8]. Each possible solution component c_i^j is expressed as the weight of pheromone level and weight of heuristic information (total weight of c_i^j). Therefore, the probability of choosing c_i^j is the weight proportion of c_i^j among the total of c_i^k with all possible values of k.

In algorithm 1, the *LocalSearch* function is an optional step. It is used to improve the solutions obtained by an individual ant after the partial solution has been constructed and before updating the pheromone. As mentioned in [5], *LocalSearch* is usually included in state-of-the-art ACO algorithms.

After all the ants finished the partial solution construction in one iteration, the pheromone should be updated to increase its value to keep strong association with the good or promising solutions. There are two main steps:

- Decrease all the pheromone values through *pheromone evaporation.*
- Increase the pheromone levels associated with a chosen set of good solutions.

Usually, most ACO algorithms use the following pheromone update rule:

$$\tau_i^j \leftarrow (1-\rho)\cdot\tau_i^j + \frac{\rho}{\mathfrak{S}_{upd}}\cdot\sum_{\{\mathfrak{s}\in\mathfrak{S}_{upd}|c_i^j\in\mathfrak{s}\}} F(\mathfrak{s}) \tag{2}$$

In the equation, $i = 1, 2, ..., n$ and $j \in |D_i|$, $|D_i|$ represents all the connected vertices with i (have accessible paths to i). The parameter ρ is an evaporation rate, with each update of the pheromone, the percentage of ρ of former pheromone will be deleted. The pheromone evaporation can avoid ACO algorithm converging too quickly. $\mathfrak{S}_{upd}$ is the subset of $\mathfrak{S}_{iter}$, and $\mathfrak{S}_{iter}\bigcup\mathfrak{s}_{bs}$ contains all the $\mathfrak{s}^p$ created by each ant in the current iteration. $\mathfrak{s}_{bs}$ represents the best-so-far solution. $F(\mathfrak{s})$ is usually called a quality function.

2.3 Main ACO Algorithm Implementations

During the past several decades, lots of different ACO algorithms have been proposed, based on our knowledge, there are three kinds of main ACO algorithm implementations: Ant System (AS), MAX-MIN Ant System (MMAS), Ant Colony System (ACS). We will give detailed explanations about these three main ACO algorithms.

2.3.1 Ant System (AS)

Ant System (AS) is the first proposed ACO algorithm in the world[8], and has been successfully applied into the famous Traveling Salesman Problem. Similar with the basis of ACO we introduced in section 2.2, AS updates pheromone in each iteration based on the solution set constructed by all the ants in each iteration. By visualizing AS as a connected construction graph (refer to Fig.3), for a vertex pair (i, j) and edge (i, j), we can re-define the pheromone τ_i^j as follows:

$$\tau_i^j \leftarrow (1-\rho)\cdot\tau_i^j + \sum_{k=1}^{n_a}\Delta\tau_{ij}^k \tag{3}$$

$$\Delta\tau_{ij}^k = \begin{cases} \dfrac{Q}{L_k} & \text{if ant } k \text{ used edge } (i,\ j) \text{ as part of its path} \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

$$p_{ij}^k = \begin{cases} \dfrac{\tau_{ij}^\alpha \cdot \eta_{ij}^\beta}{\sum_{c_{il} \in N(\mathfrak{s}^p)} \tau_{il}^\alpha \cdot \eta_{il}^\beta} & \text{if } c_{ij} \in N(\mathfrak{s}^p) \\ 0 \quad \text{otherwise} \end{cases} \tag{5}$$

In the equation (3), ρ is the evaporation rate, n_a is the total number of artificial ants, $\Delta\tau_{ij}^k$ is the quantity of pheromone left on edge (i, j) by ant k. Equation (4) gives the detail of how to calculate $\Delta\tau_{ij}^k$. Q is a constant, and L_k is the total length of all the edges that traveled by ant k in the current iteration. By having the τ_i^j value, the probability for ant k to go to vertex j from i is calculated as shown in equation (5). $N(\mathfrak{s}^p)$ is the set of feasible solution components that haven't been added into the solution set yet, if visualized as graph, it can be treated as the unvisited edges by ant k. The heuristic information $\eta_{ij} = \dfrac{1}{d_{ij}}$, where d_{ij} is the length of edge (i, j).

Therefore, similar to the process we introduced in Section 2.2, in each iteration, based on the paths traveled by each ant, the pheromone τ_{ij} on each edge of the connected construction graph is updated using equation (3). Then based on the updated pheromone, next round iteration begins, ants will travel through the graph again probabilistically choosing next vertex by using equation (5).

Based on the concept of AS, the authors also proposed an algorithm called *ant-cycle* to apply to the famous Travelling Salesman Problem (TSP). In their proposed algorithm, they set a terminating threshold NC_{max} to terminate the iteration at last to get the final optimal solution set.

2.3.2 MAX-MIN Ant System (MMAS)

MAX-MIN Ant System is proposed in the year 2000 [29], which is an improved ACO algorithm based on the original AS. MMAS differs from AS mainly in the following three aspects.

- After one iteration, only one single ant updates pheromone τ. This ant is the one who found the best solution in the current iteration (*iteration-best ant*) or the one who found the best solution from the beginning (*global-best / best-so-far ant*);

- The range of possible pheromone values on each solution component is limited to an interval $[\tau_{min}, \tau_{max}]$;
- Initialize the pheromone values to τ_{max}.

Based on the first different point listed above, the equation for updating pheromone τ_{ij} is modified as follows:

$$\tau_{ij} \leftarrow (1-\rho) \cdot \tau_i^j + \sum_{k=1}^{n_a} \Delta\tau_{ij}^{best} \tag{6}$$

By setting the range of $[\tau_{min}, \tau_{max}]$, if the updated τ_{ij} is greater than τ_{max}, then the value should be set to τ_{max}. Also, if the updated τ_{ij} is smaller than τ_{min}, the value should be set to τ_{min}. And $\Delta\tau_{ij}^{best}$ is decided by $\dfrac{1}{L_{best}}$, if edge (i, j) belongs to the path at the current iteration.

The author of MMAS claims that the main contribution they made is to utilize the pheromone limits to prevent premature convergence. Using the range of $[\tau_{min}, \tau_{max}]$ can prevent some edges have too large pheromone value, which can cause the solution convergence too soon so that the final solution may not be the best global solution. Not only MMAS, but also some other improved ACO algorithms also utilize the idea of only updating the pheromone by the single ant which found the best solution in current single iteration.

2.3.3 Ant Colony System (ACS)

The Ant Colony System (ACS) was first proposed in 1996 [13], it is also proposed by the same author (the author that proposed the concept of Ant Colony Optimization in his Ph.D. thesis at 1992. [4]). The proposed ACS algorithm has 4 phases:

- Phase 1: initialize the pheromone value to τ_0, place each artificial ant k based on some pre-defined policies to different vertices of the construction graph;
- Phase 2: execute in a cycle, in each round of the cycle, an artificial ant k makes a move from vertex i to vertex j and update the τ_{ij} based on the following formula:

$$\tau_{ij} \leftarrow (1-\rho) \cdot \tau_i^j + \rho \cdot \tau_0 \tag{7}$$

- Phase 3: the artificial ants which achieved the path L_{best} computes the delayed reinforcements $(L_{best-iter})^{-1}$, and updating τ_{ij} as follows:

$$\tau_{ij} \leftarrow (1-\alpha) \cdot \tau_i^j + \alpha \cdot (L_{best-iter})^{-1} \tag{8}$$

- Phase 4: check whether the termination condition is satisfied, if not, jump back to Phase 2.

As we can see in the 4 phases of the ACS algorithm, the main difference between AS and ACS is that ACS updates the pheromone value τ_{ij} for the edge (i, j) in each move (each construction step) of the artificial ants (move from one vertex to another vertex), not like AS, only updates τ_{ij} at the end of one construction process after one iteration was done (all the artificial ants finished their visiting to all vertices). We call the updates in each construction step as *local pheromone update*, and the updates after one iteration is called *offline pheromone update*. The main goal of the local update is to diversify the search performed by subsequent ants inside the current iteration. It can be used to decrease the pheromone concentration on some specific traversed edges so that the probability of several ants create identical paths during one iteration is lower.

2.4 Recent Researches about ACO

Since the year 1992 when Macro Dorigo proposed the first concept of ACO, the interest of digging into ACO problem has risen gradually. ACO has been applied into the full range of different research topics. In the following subsections, we will introduce some current research branches utilizing ACO.

2.4.1 Traveling Salesman Problem (TSP)

The majority of the applications of ACO is to solve NP-hard problems. The definition of NP-hard problem is the best-known algorithms to solve the problem have exponential worst case time complexity. Since ACO belongs to the category of approximate algorithms that are used to obtain good enough solutions, it is different from traditional algorithms to solve the NP-hard problem. ACO can quickly work out high-quality good enough solutions.

Traveling Salesman Problem (TSP) is one of the most famous NP-hard problems, and it is also the most common problem been studied by most of the ACO researchers over the years. Firstly, [8] described using Ant System (AS) to solve TSP; later, [13], [7], [7] mentioned using Ant Colony System(ACS) to solve TSP. TSP can be defined as follows:

Given a list of cities and distances between each pair of cities, find the shortest route that travels to each given cities and finally returns back to the original city.

It is a typical combinatorial optimization problem. We can refer to Fig. 3, each vertices in the construction graph can be treated as a city; the weighted edges can be treated as the distances between cities. If using ACS to solve TSP, the four phases listed in section 3.3 should be followed. For the Combinatorial Optimization Problem $\mathcal{P} = (\mathcal{S}, \Omega, f)$, $X_i \in \mathcal{S}$ represent the cities in the graph, the solution component c_i^j means choosing the path from city i to city j. Also, the pheromone value τ_{ij} means the level of pheromone information left on the path from city i to city j. And the probability of choosing next vertex for an artificial ant is computing using equation (5) and the pheromone value is updated using equation (7), (8) if using ACS, or equation (3), (4) if using AS.

In [7], the authors compared the performance of ACS with other naturally inspired global optimization methods including simulated annealing (SA), evolutionary program- ming (EP), genetic algorithm (GA), etc. Their comparison results are shown in Fig. 3. They reported the best integer tour length (path length when finished traveled all cities and back to the original city), the best real tour length in parentheses, and number of tours / iterations required to find the best integer tour length in square brackets. We can see for the number of iterations, ACS significantly outperforms other algorithms.

2.4.2 Other Problems

The TSP can be categorized as routing problems under NP-hard problems. Apart from TSP, there are also lots of other literature described using ACO to solve other NP-hard problems. For example, the vehicle routing problem (*find a set of minimum cost routes, starting and ending at a single location and serving a number of customers, while each customer must be served exactly once.*) which also belongs to routing problems, has been studied by [27]. Their method is to split the problem into several disjoint sub-problems based on a starting solution, then using an Ant System process to solve each of them.

Moreover, assignment problems, scheduling problems, subset problems are also studied by researchers using the concept of ACO.

CHAPTER 3

Robot Path Planning based on Ant Colony Optimization Algorithm

3.1 Modeling of Robot Motion Environment

The establishment of robot motion environmental model is a very important part of robot path planning. The actual working environment of the robot is a realistic physical space, however, the space that robot path planning algorithm to deal with is an abstract environmental space. Environmental modeling is one mapping from physical space to abstract space.

This section uses the grid method to establish the environmental model in order to simulate the actual working space of the robot. Using grids to represent maps of the robot working space can avoid complex calculations when dealing with obstacle boundaries. In the application of grid method, the division of the grid size is critical: the smaller of the grid size, the more accurate the representation of the obstacles, but at the same time, a huge amount of storage is required and the search range of the algorithm will increase exponentially; However, the larger of the grid size, the planned path can not be accurate enough. In this thesis, the grid size is $1cm * 1cm$.

3.1.1 Robot Motion Environment division using the Grid Method

For any shape of two-dimensional topographic, there are always limited number of obstacles, the coordinate location of these obstructions are easily mapped, so these can be regarded as known environmental information. Grid map can be established after traverse learning of two-dimensional space.

Without regard to robot information in height direction, two-dimensional space robots work is marked as AS, whose inner distributes limited static obstacles [35]. The size and location of the obstacles have been known, and they will not change in the motion process. In AS, the left bottom in Fig.5 is regarded as coordinates origin, the level right as X axis positive direction, the vertical upward as Y axis positive direction to establish Cartesian coordinate system $\sum$, respectively, and the maximum of the X axis and Y axis are X_{max} and Y_{max}.

Take the walk step length of the robot is δ. The step δ is used to divide X axis and Y axis regularly to get grids respectively. The number of grids in each line is $N_x = X_{max}/\delta$, and the number of grids in each row is $N_y = Y_{max}/\delta$. If there is no obstacle in a grid, this grid will be called free grid and filled with white, or it will be called obstacles grid and filled with grey.

When X_{max} is equal to Y_{max} (both are equal to 10), and δ is 1, the grid model of the robot work space is shown in Fig.5.

We adopt two main methods for grid making, named Rectangular coordinate method and Serial number method.

- Serial number method: From the left bottom of the grid map, coding the grid from bottom to top, from left to right in Fig.5. The serial number is from 1 to 100. The grid using serial number method to mark is g_n, e.g., the grid of serial number 1 is marked as g_1.

- Rectangular coordinate method Indicate every grid coordinate with central point (x, y) in Fig.5. The grid using Rectangular coordinate method is $g(x, y)$, e.g., the grid of serial number 1 is marked as $g_(0.5, 0.5)$.

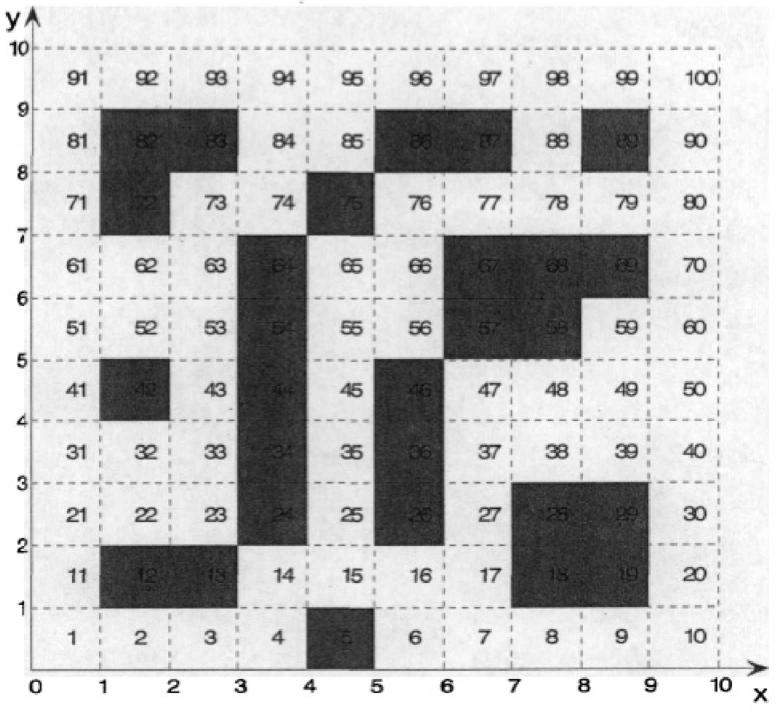


FIGURE 5: The example of grid environmental map[10].

There is a conversion relation between serial number $s_i (i = 1, 2, 3, ..., 100)$ and its coordinates:

$$\begin{cases} x_i = mod(s_i - 1, 10) + 0.5 \\ y_i = int((s_i - 1)/10) + 0.5 \end{cases} \tag{1}$$

,where *mod* represents remainder operation and *int* represents rounding operation.

In my thesis, we define the starting point of robot path planning as g_1 in the left bottom of the map, likewise, the target point of robot path planning as g_{100}.

3.1.2 The Description and Definition of Path Planning Problem

To simulate the real ant colony seeking food behavior, we assume that the starting point of robot path planning g_1 as ant colony nest, and the target point g_n as food source. Robot path planning based on ant colony optimization algorithm is actually finding an optimum path between colony nest and food source through ant colony's mutual effect and cooperation to avoid all obstacles. We make several definitions to explain the following sections conveniently.

- Definition 1: $city = 1, 2, ..., n$ represents the set of all grids, n is the total amount of girds, $n = 100$ in Fig.5;
- Definition 2: $ant = 1, 2, ..., m$ represents the set of all ants, $m = 10$ in this algorithm of my thesis;
- Definition 3: $DISTANCE_{n*n}$ is a matrix that records the distance between each grid. $DISTANCE(i, j)$ represents the distance between g_i and g_j, we have this equation:

$$DISTANCE(i,j) = \sqrt{[a(i) - a(j)]^2 + [b(i) - b(j)]^2} \tag{2}$$

  , $a(i)$ and $a(j)$ are the X-axis of the $g(i)$ and $g(j)$ respectively, likewise, $b(i)$ and $b(j)$ are the Y-axis of the $g(i)$ and $g(j)$ respectively.

3.2 Robot Path Planning based on Ant Colony Optimization Algorithm (ACA)

3.2.1 Several Changes of ACA when Solving Robot Path Planning

The mathematical model of ACA has solved the TSP problem successfully. Before applying ACA on the field of robot path planning, we should make several modifications on ACA based on the features of robot path planning.

Using Pseudo-random-proportional Rule instead of Random-proportional Rule to Choose Path

In Ant System introduced in section 2.3, we have the following state transition rule, called a random-proportional rule, is given by (3), which gives the probability with which ant k in city i chooses to move to the city j.

$$p_{ij}^{k} = \begin{cases} \dfrac{\tau_{ij}^{\alpha} \cdot \eta_{ij}^{\beta}}{\sum_{c_{il} \in N(\mathfrak{s}^{p})} \tau_{il}^{\alpha} \cdot \eta_{il}^{\beta}} & \text{if } c_{ij} \in N(\mathfrak{s}^{p}) \\ 0 \quad \text{otherwise} \end{cases} \tag{3}$$

Obviously, the ant choose the next path relied heavily on probability under random-proportional rule. To make the most of the heuristic information between adjacent nodes and pheromone value existed on each path, we decide to use a new state transition rule called pseudo-random-proportional rule, given by (4), instead of the previous random-proportional rule.

$$s = \begin{cases} \text{argmax}\{\tau_{il}^{\alpha} \cdot \eta_{il}^{\beta}\} & \text{if } q \leq q_0 \text{ (exploitation)} \\ \mathcal{S} \quad \text{otherwise (biased exploration)} \end{cases} \tag{4}$$

, where q is a random number distributed in $[0, 1]$, q_0 is a parameter, $0 \leq q_0 \leq 1$, and $\mathcal{S}$ is a random variable selected according to the probability distribution given in (3). The parameter q_0 determines the relative importance of exploitation versus exploration: every time an ant in city i has to choose a city j to move to, it samples a random number q. If $q \leq q_0$, then the best edge, according to (4), is chosen (exploitation), otherwise an edge is chosen according to (3) (biased exploration).

Redefine of Heuristic Information η_{ij}

In Ant System (AS), $\eta_{ij} = \dfrac{1}{d_{ij}}$, and d_{ij} is the distance between grid g_i and grid g_j. However, in the robot motion environment introduced in this thesis, d_{ij} could be 1 or $\sqrt{2}$ so that the function of heuristic search is not obvious and has few differences. To increase the probability of ants choosing the next grid that closer to the destination grid g_n, we redefine the heuristic information, given by (5):

$$\eta_{ij} = \frac{C}{d_{jn}} \tag{5}$$

, where C is a constant and d_{jn} is the distance between next grid g_j and the destination grid g_n.

3.2.2 The Implementation Steps of ACA on Robot Path Planning

Under grid map environment which has n grids, The implementation steps of ACA on Robot Path Planning are shown as follows:

Step 1: Set the initial parameters, initialize ant colony

The parameters including start grid g_1, destination grid g_n, iteration times $N_c = 0$, max iteration times N_{c_max}, impact index of pheromones α, impact index of heuristic factor β, pheromone evaporation rate ρ, constant Q, and initial pheromone $\tau_{ij}(0) = const$.

Set $routh\_best$ to record the shortest path in every iteration, $length\_best$ to record the length of the shortest path in every iteration, $length\_average$ to record the average length of all paths got in each iteration.

All the ants m are placed in the start grid g_1, and put g_1 into the taboo table of the ant k, $tabu_k(k = 1, 2, 3, ...m)$;

Step 2: Each ant move into next grid

According to pseudo-random-proportional rule given by (3) and (4), each ant selects a next grid to move in, and the next grid will put into the taboo table of this ant. If the current grid is the destination grid, let this ant dead.

Step 3: Repeat step 2, until all the ants have selected the next grid

Step 4: Local pheromone update

After all ants have selected the next grid, the algorithm implements local pheromone updating given by (6).

$$\tau_{ij}(n+1) = (1-\rho) \cdot \tau_{ij}(n) + \rho \cdot \Delta\tau_{ij}^k \tag{6}$$

, where $\Delta\tau_{ij}^k$ is a constant, $\tau_{min} < \Delta\tau_{ij}^k < \tau_{max}$. When $\tau_{ij}(n+1) < \tau_{min}$, set $\tau_{ij}(n+1) = \tau_{min}$; When $\tau_{ij}(n+1) > \tau_{max}$, set $\tau_{ij}(n+1) = \tau_{max}$

Step 5: Repeat Step 2, Step 3, and Step 4, until all the ant move into the destination grid

Step 6: Calculate the optimal path, the length of the optimal path and the average length of all path ants walk in this literation

Step 7: Global pheromone update

After each iteration, the algorithm implements global pheromone updating given by (7) and (8).

$$\tau(r,s) \leftarrow (1-\alpha)\tau(r,s) + \alpha\Delta\tau(r,s) \tag{7}$$

$$\Delta\tau(r,s) = \begin{cases} \dfrac{1}{L_{best-iter}} & \text{if (r,s) belongs to the global optimal path} \\ 0 \quad \text{otherwise} \end{cases} \tag{8}$$

Step 8:Clear the $tabu_k$, $N_c = N_c + 1$, if $N_c \leq N_{c\_max}$, move to Step 2; if $N_c \leq N_{c\_max}$, move out the whole iteration, get the optimal path and the length of the optimal path

CHAPTER 4

Robot Path Planning based on ACA integrating with Genetic Algorithm (GA)

4.1 Genetic Algorithm and Robot Path Planning

4.1.1 Introduction of Genetic Algorithm

Genetic Algorithms became popular through the work of John Holland in the early 1970s. A Genetic Algorithm (GA) is a meta-heuristic inspired by the process of natural selection that belongs to evolutionary algorithms (EA) [24].

In a genetic algorithm, an optimization problem can be simplified to a process to find a better solution from a population of candidate solutions. Each candidate solution has a set of properties which can be mutated and altered; traditionally, solutions are represented in binary as strings of 0s and 1s, but other encodings are also possible.

A typical genetic algorithm requires a genetic representation of the solution domain and a fitness function to evaluate the solution domain. A standard representation of each candidate solution is as an array of bits. The main property that makes these genetic representations convenient is that their parts are easily aligned due to their fixed size, which facilitates simple crossover operations [34]. Variable length representations may also be used, but crossover implementation is more complex in

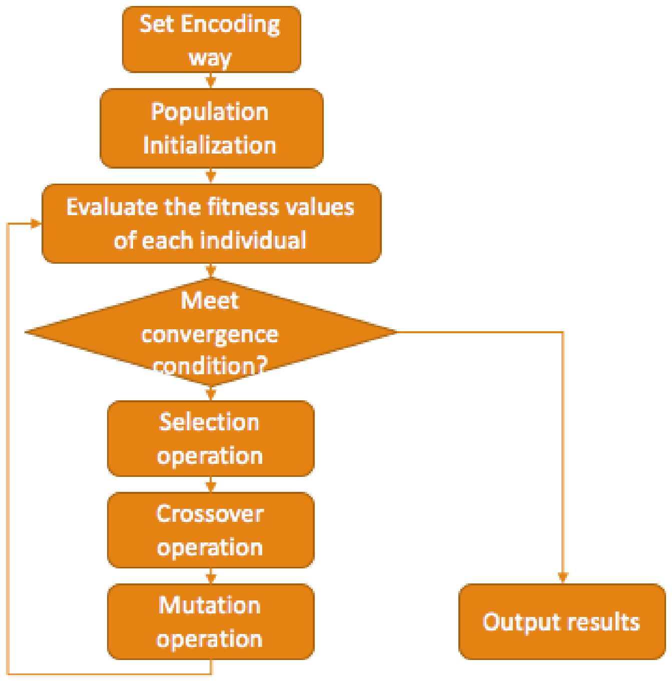

Set Encoding way
Population Initialization
Evaluate the fitness values of each individual
Meet convergence condition?
Selection operation
Crossover operation
Mutation operation
Output results

FIGURE 6: The flow chart of genetic algorithm.

this case. Once the genetic representation and the fitness function are defined, a GA proceeds to initialize a population of solutions and then to improve it through repetitive application of the mutation, crossover, inversion and selection operators.

In summary, the core content of genetic algorithm involves the generation of initial population, the selection of fitness function, the design of genetic operators and termination conditions. The flow chart of GA is shown in Fig.6.

4.1.2 The Implementation Steps of GA on Robot Path Planning

Due to the capability of rapid global search and rapid search to global optimal path that genetic algorithms have, genetic algorithms have been applied to multi-robot path planning. In this section, it will adopt genetic algorithm to find the optimal path based on the grid environmental map introduced in chapter three. The implementation steps of GA on Robot Path Planning are shown as follows:

Step 1: Path encoding

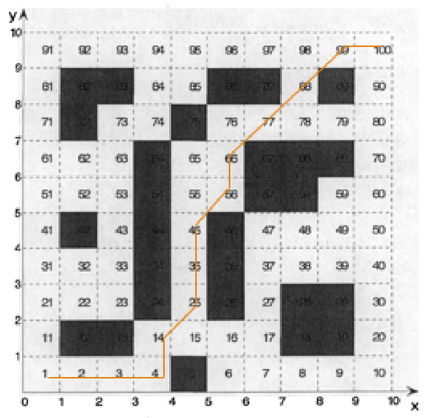


FIGURE 7: The example of path encoding.

Binary coding or floating point coding is not suitable when using GA to solve robot path planning problems. This thesis adopts a series of grid numbers as path encoding, which means each path can be represented as a series of numbers. Due to one path can not pass the obstacle grids, nor can it pass through the repeated grids, this series of numbers can not be the serial number of obstacle girds or repeated girds. For example, one path can be represented as follows in Fig.7:

1-2-3-4-14-25-35-45-56-66-77-88-99-100

The path is identified with orange line, and the figure 1 represents the serial number of path planning's starting grid, likewise, the figure 100 represents the serial number of path planning's target grid. The rest of numbers represent the serial number of path planning's middle grids.

Step 2: The generation of initial population

In the grid environment model, the path from the starting grid to the target grid is variable, so in the genetic algorithm, the chromosome representing the path is also variable. The initial path individual generation process is as follows: starting from the starting grid 1, step by step, non-repeated to choose next free grid until moves to the target grid 100. Therefore, the path population consist of the multiple path individuals that gained from step 2.

Step 3: The selection of fitness function

The fitness function is the most important factor to evaluate the convergence and stability of the genetic algorithm. As robot path planning problem has to satisfy the condition of the shortest path, the fitness function set the path length as the evaluation criterion. The fitness function in this thesis is shown as follows [35]:

$$f = \frac{1}{(1 + \frac{1}{\sqrt{N-1}})d} \tag{1}$$

where, N represents the number of passed grids in each path, d represents the length of each path.

Step 4: The design of genetic operators

This thesis introduces four genetic operators: selection operator, crossover operator, mutation operator, deletion operator.

- Seletion operator:

  The selection operation is also called copy operation, which means it selects the individual from the parent and passes individual to the offspring instantly. In the process of copy, the probability that an individual in each parent copied to the next generation is determined by the value of its fitness function. Individuals with larger fitness function value, ie, individuals with shorter paths, are more likely to be copied to the next generation. So how to select individuals to copy to the next generation; this thesis uses roulette method. To illustrate the working principle of roulette method, assuming that the population has path 1, path 2 and path 3 three individuals, with the fitness function value 2, 3, 5, respectively. The probability of these three individuals to be selected is [0.2, 0.3, 0.5], and cumulative probability is [0.2, 0.5, 1]. Generate a random number *rand* between 0 to 1, if *rand* falls in [0, 0.2], path 1 is selected; if *rand* falls in [0.2, 0.5], path 2 is selected; if *rand* falls in [0.5, 1], path 3 is selected.

- Crossover operator:

  The crossover operation is to intersect different individuals from the parent to produce new individuals. The typical crossover methods have single point crossover, double point cross and multipoint crossover. This thesis uses single point crossover method. According to the given crossover probability, we select two path individuals from the parent to cross and exchange the part of the path afer the intersection point.

For example, One path is: *1-4-14-25-45-56-66-99-100*, another path is *1-4-14-15-65-66-76-80-100*, the crossover grid is 66, then two path will be *1-4-14-25-45-56-66-76-80-100* and *1-4-14-15-65-66-99-100*, respectively.

- Mutation operator:

  Mutation operation plays a key role in increasing population diversity. Considering the diversity of the paths after searching, the path individuals after mutation operation is not considered to be superior to the path individuals before mutation operation. According to a given mutation probability, we select a serial number of grid from one path individual randomly and replace an arbitrary serial number with the selected serial number of the grid. For example, one path is *1-4-14-25-45-56-66-99-100*, if the mutation grid is 56, if we choose 55 to replace that grid, the new path after mutation will be *1-4-14-25-45-55-66-99-100*,

- Deletion operator:

  Due to the initial randomness and mutation operations, the individuals from the path population may have an obstacle grid number, and the path individual can not contain an obstacle grid number, which requires the removal of obstacles in the path individual. When implementing crossover and mutation operations, the path individuals may contain some repeated grid numbers, which is not allowed, It is necessary to delete the serial numbers of which two repeated grids and one of the two repeated grids.

4.2 Robot path planning based on GA-ACA

4.2.1 Introduction of GA-ACA

Genetic algorithm (GA) is a global optimal algorithm based on natural selection and natural genetic, with the capability of rapid global search and rapid search to global optimal path. However, without the use of the feedback information of the system, this method usually cause redundancy iteration and reduce solution efficiency. Ant colony optimization algorithm (ACA) converge to the optimal path through pheromone accumulation and update, with distributivity, parallelism and global convergence ability, but in the initial stage pheromone among all paths are equal, which makes it equate to greedy algorithm and leads to slow convergence speed, the obtained solution is often not the optimal solution.

To overcome the drawback of two algorithms in robot path planning application, a method of path planning was put forward, which combined genetic algorithm with ant colony optimization algorithm, called GA-ACA. GA-ACA algorithm is firstly using genetic algorithm to generate distributed initialization pheromone, then using ant colony algorithm for the optimal solution, thus this way effectively combines fast convergence of genetic algorithm and information positive feedback mechanism of ant colony algorithm.

GA-ACA is superior to the genetic algorithm in computational efficiency, and is superior to the ant colony optimization algorithm in time efficiency.This algorithm has become a kind of heuristic algorithm which has better computational efficiency and time efficiency.

4.2.2 The Implementation Steps of GA-ACA on Robot Path Planning

Under grid map environment which has n grids, The implementation steps of GA-ACA on Robot Path Planning are shown as follows:

Step 1: We adopt a series of grid numbers as path encoding, which means each path can be represented a series of numbers. The series of numbers can not be the serial number of obstacle grids or repeated grids. Then, we generate initial path population and select the fitness function related to the path length.

Step 2: We calculate the value of fitness function for each path individual from the path population and use the roulette method to select the path individuals that will be done by crossover and mutation operations.

Step 3: According to the given crossover rate P_c, we select two path individuals to do the crossover operation. The specfic method is to generate a random number $rand$ between 0 to 1, if $rand < P_c$, do the crossover operation, otherwise, don't operate.

Step 4: According to the given mutation rate P_m, we select two path individuals to do the mutation operation. The specfic method is to generate a random number $rand$ between 0 to 1, if $rand < P_m$, do the mutation operation, otherwise, don't operate.

Step 5: Repeat Step 2 to Step 4, until satisfies the given convergence condition and iteration times and generate several optimized path individuals.

Step 6: Generate distributed initialization pheromone according to the optimized path individuals. Next, set the initial parameters of the ant colony algorithm. All the ants m are placed in the start grid g_1, and put g_1 into the taboo table of the ant k, $tabu_k(k = 1, 2, 3, ...m)$;

Step 7: Each ant selects a next grid to move in according to the state transition rule, and the next grid will be put into the taboo table of this ant.

Step 8: Repeat Step 7, until this ant constructs a completive path, and pheromone is updated locally.

Step 9: Repeat Step 7 and Step 8, until all the ants complete their paths respectively, and pheromone is updated globally.

Step 10: Remove the taboo table of all ants, repeat from Step 7 to Step 9, until reach the set number of cycles or meet certain termination condition.

Step 11: Output the optimal path.

The flow chart of GA-ACA is shown as Fig.8.

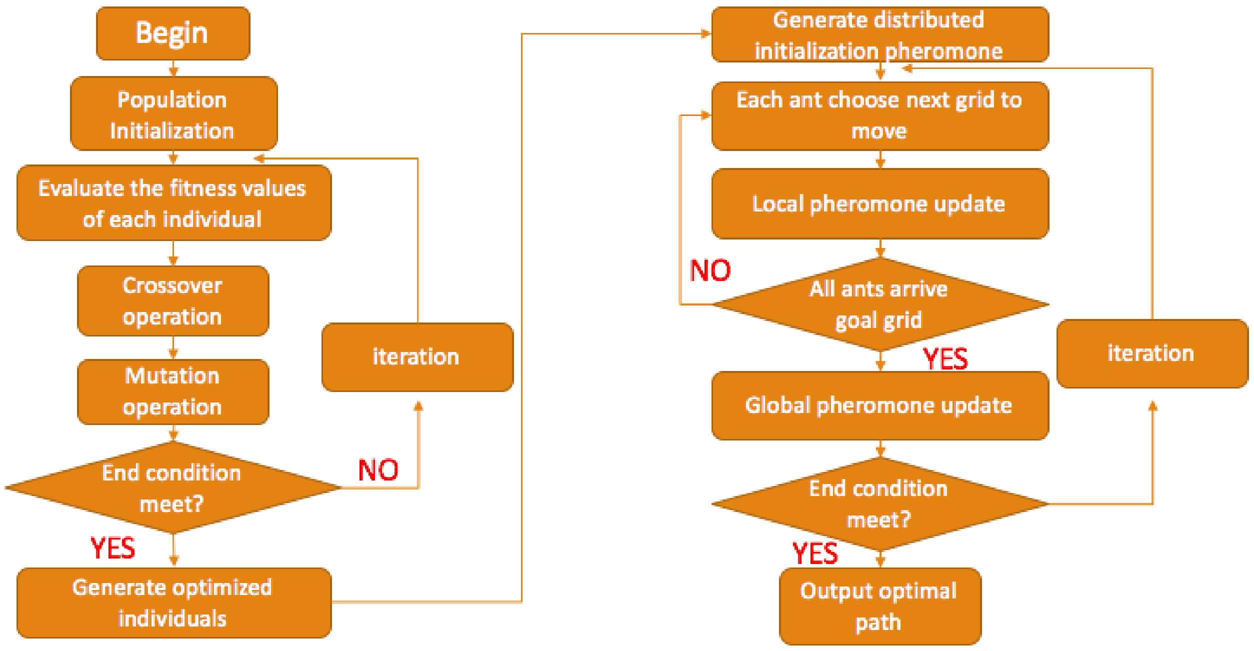

Begin
Population Initialization
Evaluate the fitness values of each individual
Crossover operation
Mutation operation
End condition meet?
NO
iteration
YES
Generate optimized individuals
Generate distributed initialization pheromone
Each ant choose next grid to move
Local pheromone update
NO
All ants arrive goal grid
YES
Global pheromone update
iteration
End condition meet?
YES
Output optimal path

FIGURE 8: The flow chart of GA-ACA.

4.3 Robot path planning based on ACA-GA

4.3.1 Introduction of ACA-GA

The GA-ACA algorithm introduced in 4.2 apply GA and ACA in two stages essentially, rather than an integration of these two algorithms in the true sense. In this section, we will integrate into solution of ant colony optimization algorithm an idea of crossover about genetic algorithm. The main framework of this algorithm is ACA, but it integrates with GA during the intermediate solution process. To distinguish this new algorithm with GA-ACA, we call it ACA-GA.

In short, ACA-GA selects two paths randomly after one iteration, and do the crossover operation on two paths, and update pheromone of new path if it is better than the optimal path in current iteration.

The crossover operation in the genetic algorithm is introduced into the ant colony optimization algorithm, which can increase the diversity of the solution and speed up the problem-solving speed.

4.3.2 The Implementation Steps of ACA-GA on Robot Path Planning

Under grid map environment which has n grids, The implementation steps of ACA-GA on Robot Path Planning are shown as follows:

Step1: Set the initial parameters of the ant colony algorithm. All the ants m are placed in the start grid g_1, and put g_1 into the taboo table of the ant k, $tabu_k(k = 1, 2, 3, ...m)$;

Step2: According to the state transition rules, each ant selects a next grid to move in, and the next grid will be put into the taboo table of this ant.

Step3: Repeat Step2, until the ant constructs a completive path, and pheromone is updated locally.

Step4: Repeat Step2 and Step3, until all the ants complete their paths respectively.

Step5: Select the optimal path in this iteration, then randomly choose another path of the iteration, according to the given crossover rate, the two paths are taken crossover operation. For example, under the grid environment map in Fig.5, we assume that the optimal path in one iteration is represented as:

1-2-3-14-25-35-45-55-65-66-77-88-99-100

Then, we select another path randomly, represented as follows:

1-2-3-4-14-25-35-45-56-66-77-88-99-100

We can see that the length of both two paths is 15.0711. If we select 45 as the crossover grid, the two paths after crossover operation are represented as follows respectively:

1-2-3-14-25-35-45-56-66-77-88-99-100,

1-2-3-4-14-25-35-45-55-65-66-77-88-99-100

The length of the two paths after crossover operation are 14.4853 and 15.6569 respectively. Obviously, it generates a new path better than the previous optimal path in the iteration.

Step6: Update pheromone globally and remove the taboo table of all ants, re-

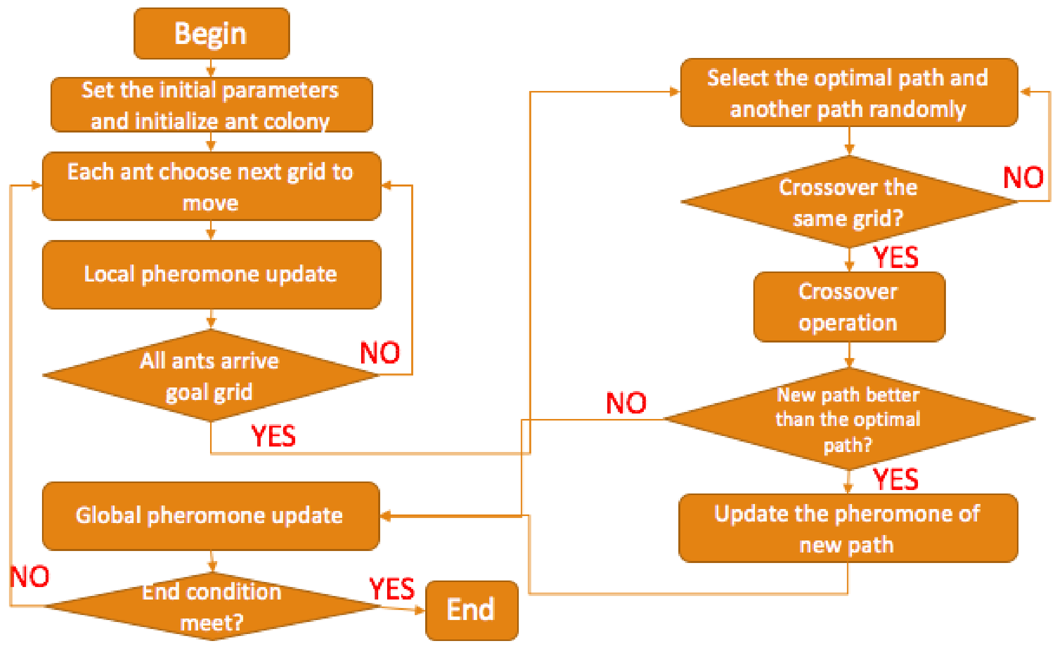

Begin
Set the initial parameters and initialize ant colony
Each ant choose next grid to move
Local pheromone update
All ants arrive goal grid
NO
YES
Global pheromone update
End condition meet?
NO
YES
End
Select the optimal path and another path randomly
Crossover the same grid?
NO
YES
Crossover operation
New path better than the optimal path?
NO
YES
Update the pheromone of new path

FIGURE 9: The flow chart of ACA-GA.

peat from Step2 to Step5, until reaching the set number of cycles or meet certain termination condition.

Step7: Output the optimal path.

The flow chart of ACA-GA is shown as Fig.9.

4.4 Performance Evaluation Indexes of Ant Colony Optimization Algorithm

In order to comprehensively measure the performance of the ant colony algorithm, the following basic indexes to evaluate the performance of ant colony algorithm are introduced below [32]:

1. **The Best Performance Index**:

   Let E_O represents the best performance index, the formula is as follows:

$$E_O = \frac{c_b - c^*}{c^*} \times 100\% \tag{2}$$

   where c_b represents the optimal value obtained by the algorithm; c^* represents the theoretical optimal value. When the theoretical optimal value is unknown, it is replaced by the best known value. The optimal performance index is used to measure the optimal optimization degree of the ant colony algorithm. The smaller of the value means that the optimal performance of the ant colony algorithm is better.

2. **The Time Performance Index**:

   Let E_T represents the time perfomance index, the formula is as follows:

   $$E_T = \frac{I_a T_0}{I_{max}} \times 100\% \tag{3}$$

   where I_a represents the algorithm's number of iterations when it meets convergence condition(In this paper, I_a refers to the iteration number when the mean path length tend to be stable); T_0 represents the average execution time of one iteration; I_{max} represents the algorithm's number of iterations. The time performance index is uesd to measure the search speed of the ant colony algorithm. The smaller of the E_T means that the convergence speed of the ant colony algorithm is quicker.

3. **The Robustness Performance Index**:

   Let E_R represents the robustness perfomance index, the formula is as follows:

   $$E_R = \frac{c_a - c^*}{c^*} \times 100\% \tag{4}$$

   where c_a represents the average path length value; c^* represents the theoretical optimal value. When the theoretical optimal value is unknown, it is replaced by the best known value. The smaller of the E_R means that the stability of the ant colony algorithm is better.

CHAPTER 5

Simulation Experiments

5.1 Previous Experiments on GA-ACA and ACA-GA

In 2010, [15] proposed the GA-ACA algorithm. The basic thought of GA-ACA algorithm is that this algorithm uses GA to generate the initial pheromone distribution in the former stage and then uses ACA to work out the final solution in the later stage. The authors applied ACA and GA-ACA on two grid environment maps with deep traps, and the simulation results indicated that GA-ACA performed better (obtain shorter path) and converged quicker than ACA when dealing with path planning problems in environment with deep traps.

In 2012, [35] proposed the ACA-GA algorithm. This algorithm which is a fast and efficient heuristic algorithm makes full use of the advantages of genetic algorithm and ant colony algorithm. The authors set obstacles in the $10 * 10$ grid environment map and compile the simulation program using VC++ 6.0 and MATLAB 7.0. GA, ACA, and ACA-GA algorithms would be done by simulation analysis respectively. The simulation results showed that the ability of ACA-GA algorithm to search the optimal solution, convergence rate and stability is higher than that of the ant colony algorithm and that of genetic algorithm [35].

However, how the GA-ACA and ACA-GA perform when dealing with path planning problems in different static environment maps? Which is better when dealing with path planning problems in a complicated environment map with traps, GA-ACA or ACA-GA? To anwser these questions, four groups of experiments will be

TABLE 1: The simulation platform source code list

Source code file	Function description
fitness.m	The fitness function
pop_generator.m	The ant population initialization
select.m	The select operation during GA process
cross.m	The cross operation during GA process
mutation.m	The mutation operation during GA process
insert_grid.m	The insertion operation during GA process
Robot_Path_Optimization_GA_Sim.m	robot path planning using GA
Robot_Path_Optimization_ACA_Sim.m	robot path planning using ACA
Robot_Path_Optimization_GA_ACA_Sim.m	robot path planning using GA_ ACA
Robot_Path_Optimization_ACA_GA_Sim.m	robot path planning using ACA_ GA

implemented in this thesis.

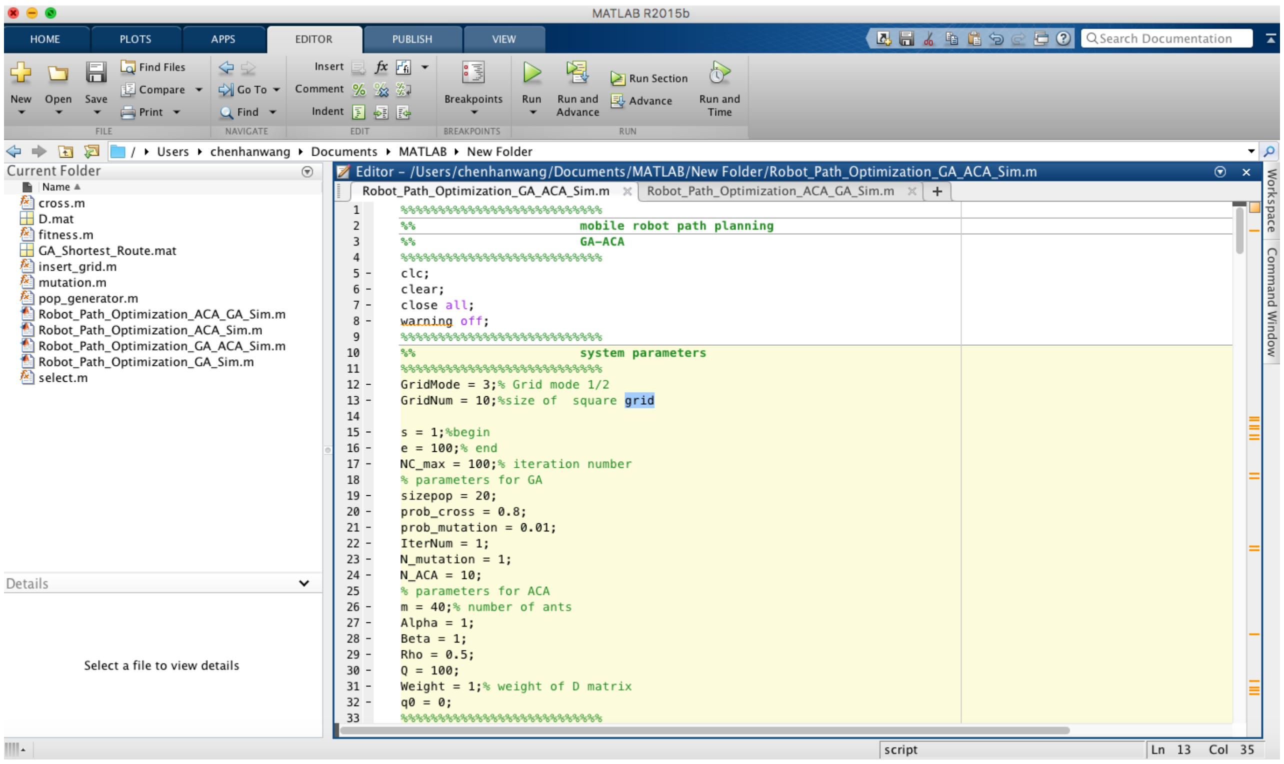
5.2 Simulation System

We compile the simulation programs using MATLAB, and GA, ACA, GA-ACA and ACA-GA algorithms had be done by simulation analysis respectively. The screenshot of the programming environment for my simulation experiments is shown in Fig.10. The simulation program source code list is shown in Table 1.

5.3 Parameter Settings

1. **The number of ants** m:

   The number of ants m has a crucial influence on the overall performance of the ant colony optimization algorithm. In general, a certain number of ants could increase the ACA's capability of global search and stability. However, if the number of ants m is oversize, which will make the pheromone variation become average and slow down the convergence speed, in contrast, if the number of ants m is too few, which will reduce the algorithm stability and lead the problems of early stagnation [10]. In [35], the authors set $m = 10$ under $10 * 10$ grid map environment. We set $m = 40$ under $10 * 10$ grid map environment in this thesis.


MATLAB R2015b
HOME
PLOTS
APPS
EDITOR
PUBLISH
VIEW
Current Folder
cross.m
D.mat
fitness.m
GA_Shortest_Route.mat
insert_grid.m
mutation.m
pop_generator.m
Robot_Path_Optimization_ACA_GA_Sim.m
Robot_Path_Optimization_ACA_Sim.m
Robot_Path_Optimization_GA_ACA_Sim.m
Robot_Path_Optimization_GA_Sim.m
select.m
Details
Select a file to view details
Editor – /Users/chenhanwang/Documents/MATLAB/New Folder/Robot_Path_Optimization_GA_ACA_Sim.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                       mobile robot path planning
%%                       GA-ACA
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clc;
clear;
close all;
warning off;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                       system parameters
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
GridMode = 3;% Grid mode 1/2
GridNum = 10;%size of  square grid

s = 1;%begin
e = 100;% end
NC_max = 100;% iteration number
% parameters for GA
sizepop = 20;
prob_cross = 0.8;
prob_mutation = 0.01;
IterNum = 1;
N_mutation = 1;
N_ACA = 10;
% parameters for ACA
m = 40;% number of ants
Alpha = 1;
Beta = 1;
Rho = 0.5;
Q = 100;
Weight = 1;% weight of D matrix
q0 = 0;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

FIGURE 10: The simulation system using MATLAB.

2. **The combination of parameters α, β and ρ:**

   The impact index of pheromones α reflects the relative importance of pheromone accumulated by ant colony and the impact index of heuristic factor β reflects the relative importance of heuristic information.

   Pheromone evaporation rate ρ reflects the intensity of the interaction between ants, directly related to the global search ability and convergence speed of ant colony optimization algorithm. By increasing ρ, we can improve the global search ability of ant colony optimization algorithm. However, it will reduce the convergence speed of the algorithm.

   In fact, the roles of α, β and ρ are related closely. When applying ant colony optimization algorithm on robot path planning, the wrong combination setting of α, β and ρ will slow down the solution speed and make the quality of results unexpectable. In this thesis, the setting of these three parameters is: $\alpha = 1$, $\beta = 0.0$, $\rho = 0.5$.

3. **The pheromone intensity Q:**

   Pheromone intensity Q is the total amount of pheromone released by the ant colony left on the paths they traveled after one iteration. The value of Q is to make full use of the global information feedback and make the algorithm search for the optimal path at a reasonable rate of evolution. The larger the Q, the faster the pheromone accumulation on the paths of the ant colony, the convergence speed of the algorithm is improved [32]. However, when Q is oversize, the algorithm easily fall into local optimal solution.

4. **The number of iteration N_c:**

   To ensure the algorithm can search the optimal path within the number of iterations, the value of N_c should be set larger. Under the $10 * 10$ grid environment map, $N_c = 100$ in this thesis.

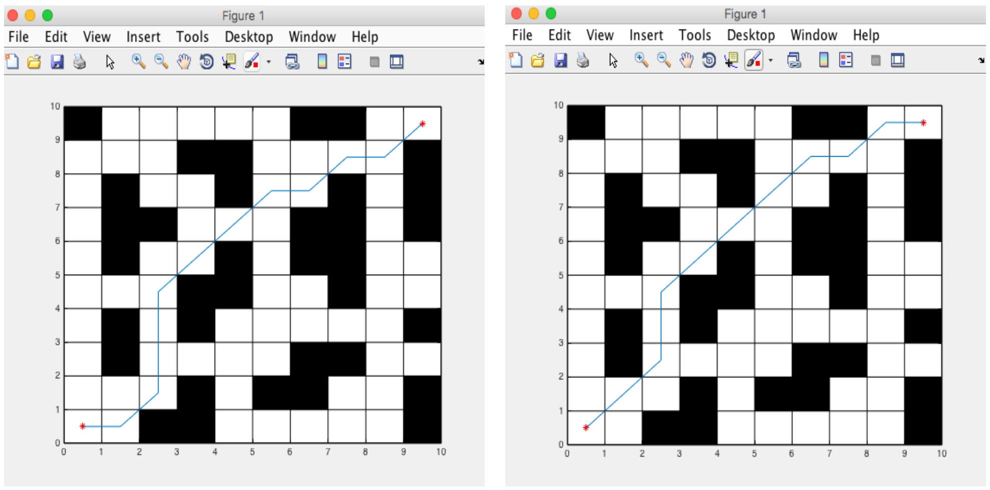
5.4 The Implementation of Simulation Experiments

5.4.1 The First Group of Experiments

To verify the effectiveness of GA-ACA and ACA-GA compared with GA and ACA under the same grid environment, we set several obstacles in the $10 * 10$ grid environment map arbitrarily (map 01 shown below in Fig.11 and Fig.12) and compile the simulation program using MATLAB. The GA, ACA, ACA-GA and GA-ACA algorithms will be compared by simulation experiments respectively. The optimal path obtained by GA and ACA in map 01 is shown in Fig.11. And the optimal path obtained by GA-ACA and ACA-GA in map 01 is shown in Fig.12.

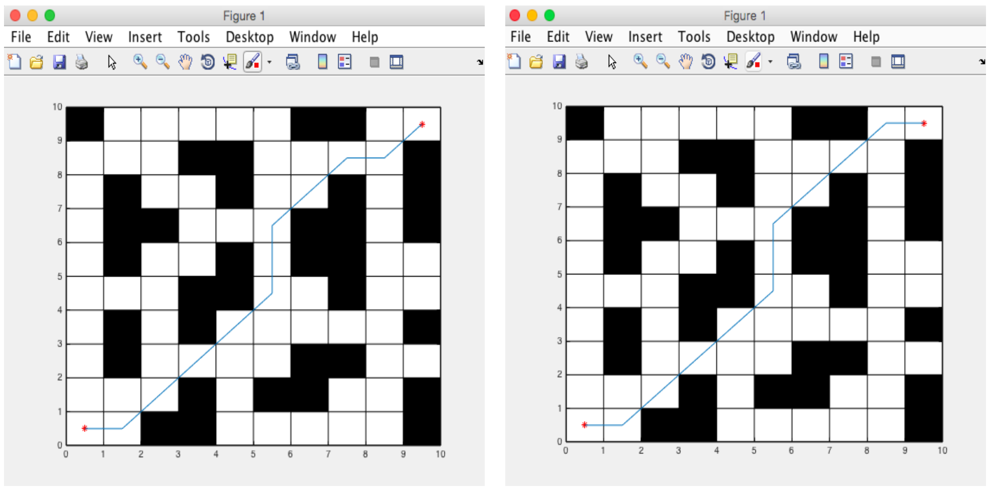
We can see that the optimal path length obtained by GA, ACA, GA-ACA and ACA-GA algorithm is 14.4853, 13.8995, 13.8995, 13.8995 respectively from Fig.11 and Fig.12, indicating that the search capability of GA-ACA and ACA-GA is better than GA, and equivalent to ACA in map 01.

In order to see the trend of fluctuations of the shortest path length from iteration 1 to 100, the optimal path length evolution graph and the mean path length evolution



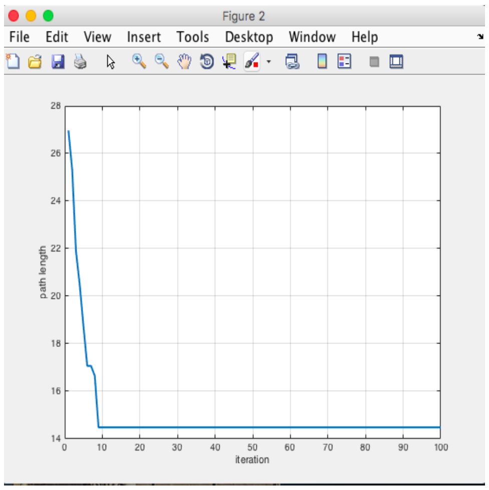
(a) (b)

FIGURE 11: (a)The optimal path obtained by GA in map 01, the length of the optimal path is 14.4853; (b)The optimal path obtained by ACA in map 01, the length of the optimal path is 13.8995.

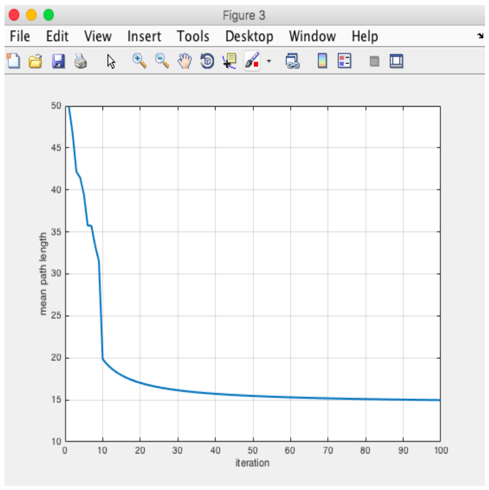


(a) (b)

FIGURE 12: (a)The optimal path obtained by GA-ACA in map 01, the length of the optimal path is 13.8995; (b)The optimal path obtained by ACA-GA in map 01, the length of the optimal path is 13.8995.

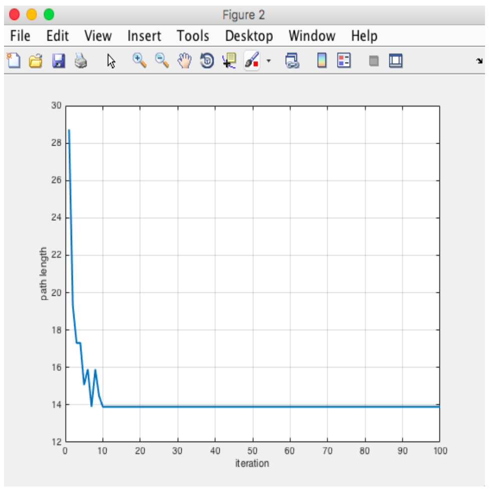


(a)

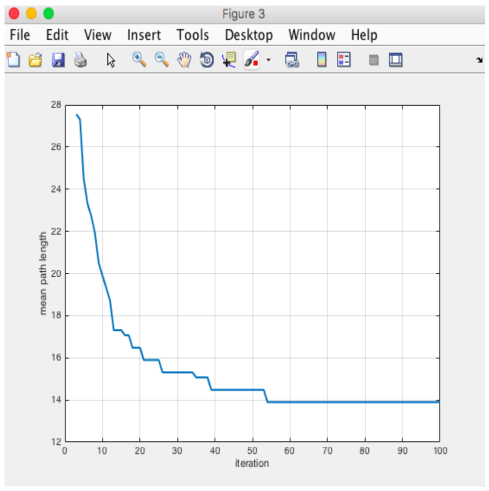


(b)

FIGURE 13: (a)The optimal path length evolution graph by using GA in map 01; (b)The mean path length evolution graph by using GA in map 01.

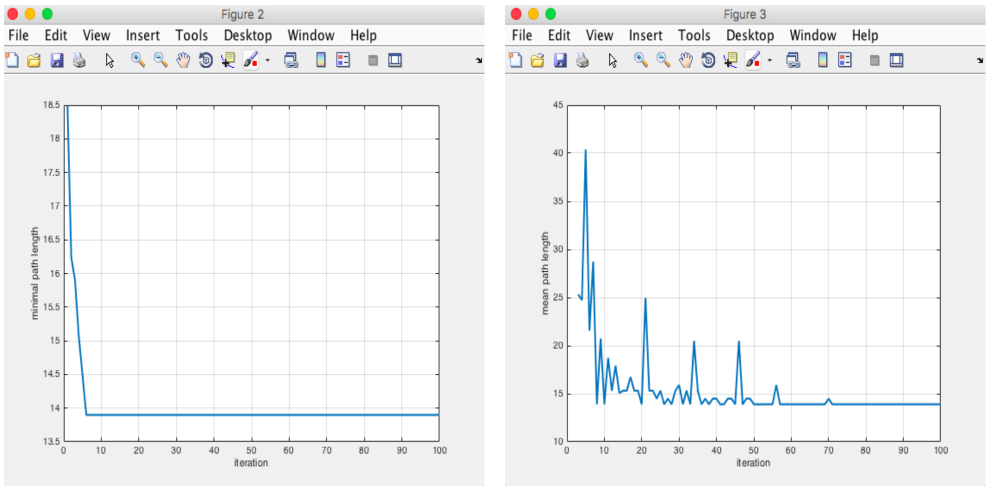


(a)



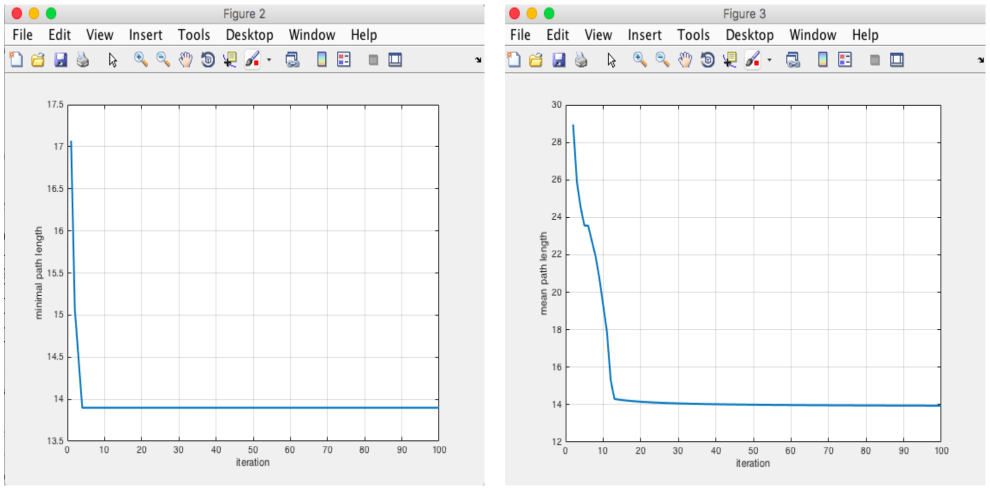
(b)

FIGURE 14: (a)The optimal path length evolution graph by using ACA in map 01; (b)The mean path length evolution graph by using ACA in map 01.



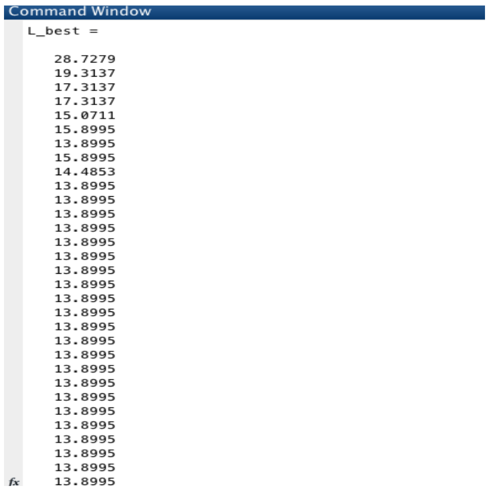
(a) (b)

FIGURE 15: (a)The optimal path length evolution graph by using GA-ACA in map 01; (b)The mean path length evolution graph by using GA-ACA in map 01.



(a) (b)

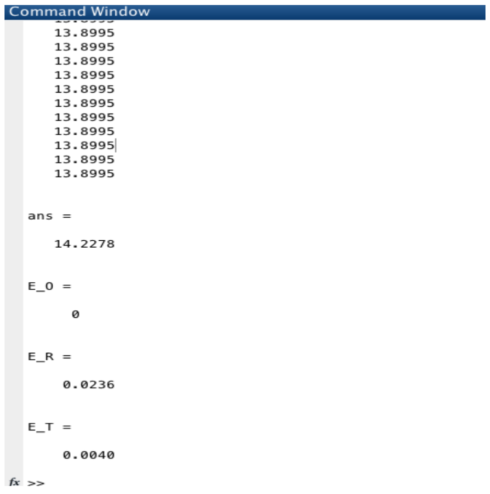
FIGURE 16: (a)The optimal path length evolution graph by using ACA-GA in map 01; (b)The mean path length evolution graph by using ACA-GA in map 01.



```
Command Window
L_best =

   28.7279
   19.3137
   17.3137
   17.3137
   15.0711
   15.8995
   13.8995
   15.8995
   14.4853
   13.8995
   13.8995
   13.8995
   13.8995
   13.8995
   13.8995
   13.8995
   13.8995
   13.8995
   13.8995
   13.8995
   13.8995
   13.8995
   13.8995
   13.8995
   13.8995
   13.8995
   13.8995
   13.8995
   13.8995
   13.8995
fx 13.8995
```

(a)



```
Command Window
   13.8995
   13.8995
   13.8995
   13.8995
   13.8995
   13.8995
   13.8995
   13.8995
   13.8995
   13.8995
   13.8995


ans =

   14.2278


E_O =

     0


E_R =

    0.0236


E_T =

    0.0040

fx >>
```

(b)

FIGURE 17: The simulation result of ACA in map 01

graph are plotted respectively for each algorithm. These graphs are shown from Fig.13 to Fig.16. The optimal path length evolution graph reflects how the length of optimal path vary from iteration 1 to 100. However, the mean path length evolution graph reflects the mean value of the path length obtained by the ant colony in every iteration. For example, there are 40 ants and each ant can find a path from the starting point to the destination point, the mean path length is the mean value of the length of all these 40 paths obtained by each ant.

The optimal path length evolution graph and the mean path length evolution graph are plotted by implementing GA (Fig.13), ACA (Fig.14), GA-ACA (Fig.15), and ACA-GA (Fig.16) respectively. In Fig.13(a), we can see that the optimal path is found at the iteration 9, and the ant colony converge to the optimal path at iteration 70 by using GA in Fig.13(b); In Fig.14(a), the optimal path is found at iteration 7, and the ant colony converge to the optimal path at iteration 53 by using ACA in Fig.14(b); In Fig.15(a), the optimal path is found at iteration 6, and the ant colony converge to the optimal path at iteration 50 by using GA-ACA in Fig.15(b); In Fig.16(a), the optimal path is found at iteration 4, and the ant colony converge to the optimal path at iteration 30 by using ACA-GA in Fig.16(b); Compared with these four algorithms, ACA-GA can converge to the optimal path at minimum iteration times.

TABLE 2: The performance index of four algorithms in Experiment one

Algorithm name	Best (%)	Timing (%)	Robustness (%)	Iteration numbers	
				Optimal found	Optimal convergence
GA	4.21	0.61	3.34	9	70
ACA	0	0.40	2.36	7	53
GA-ACA	0	0.21	0.77	6	50
ACA-GA	0	0.13	0.35	4	30

We decide to apply the best performance index, time performance index and robustness performance index introduced in Chapter 4 to evaluate these four algorithms' performance comparatively. The simulation result of ACA in map 01 is shown in Fig.17. We can clearly see that the optimal path ($L_{best} = 13.8995$) is found at iteration 7. The value of ans (14.2278) is the mean value of the optimal path length after ACA running 100 times, which refers to c_a in chapter 4.4. The value of E_O, E_R and E_T represents the best performance index, the robustness index and the time performance index respectively. The experiment results of the four different algorithms display in Table 2.

In Table 2, we can see that the best performance index of ACA, GA-ACA and ACA-GA are all equal to 0%, lower than 4.21% of GA, indicating that the ability of ACA, GA-ACA and ACA-GA to search the optimal solution is higher than that of GA in map 01.

Regarding the timing performance index, ACA-GA is 0.13%, obviously lower than that of GA, ACA and GA-ACA, which demonstrates that the convergence rate of ACA-GA is higher that the other three algorithms in map 01.

Regarding the robustness performance index, we can see that the robustness of GA-ACA(0.77%) and ACA-GA(0.35%) is obviously lower that the other traditional algorithms, GA(3.34%) and ACA(2.36%), which demonstrates that the stability of GA-ACA and ACA-GA is greater than that of GA and ACA in map 01.

5.4.2 The Second Group of Experiments

In the first group of the experiment, we can see that GA-ACA and ACA-GA perform better than GA and ACA when solving robot path planning problems. However, there is little difference between GA-ACA and ACA-GA regarding the results of time performance index and robustness performance index. In the second group of experiments, we design a grid environment map with one trap (map 02 shown in Fig.18) to compare the performance of GA-ACA and ACA-GA when dealing with grid maps which have one trap. The optimal path obtained by GA-ACA and ACA-GA in map 02 is shown in Fig.18.

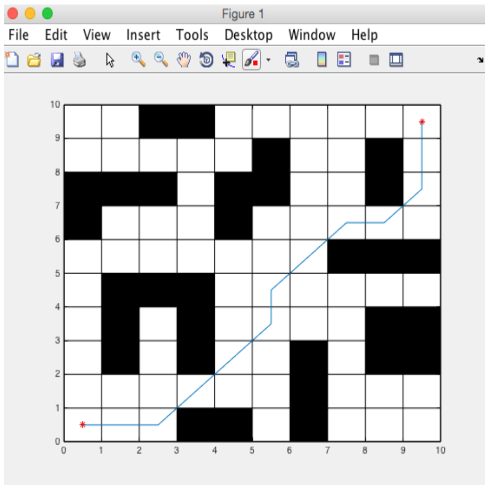
We can see that the optimal path length obtained by GA-ACA and ACA-GA algorithm is both 14.4853 from Fig.18. It may indicate that the search capability of GA-ACA and ACA-GA is the same when dealing the grid environment with one trap in the second group of experiment.

The results of the optimal path length evolution graph and the mean path length evolution graph are plotted by implementing GA-ACA (Fig.19), and ACA-GA (Fig.20) respectively. In Fig.19(a), we can see that the optimal path is found at iteration 17, and the ant colony converge to the optimal path at iteration 71 by using GA-ACA in Fig.19(b); In Fig.20(a), the optimal path is found at iteration 9, and the ant colony converge to the optimal path at iteration 50 by using ACA-GA in Fig.20(b). Compared with GA-ACA, ACA-GA can find the optimal path and converge to the optimal path quickly when dealing with grid maps contains one trap.

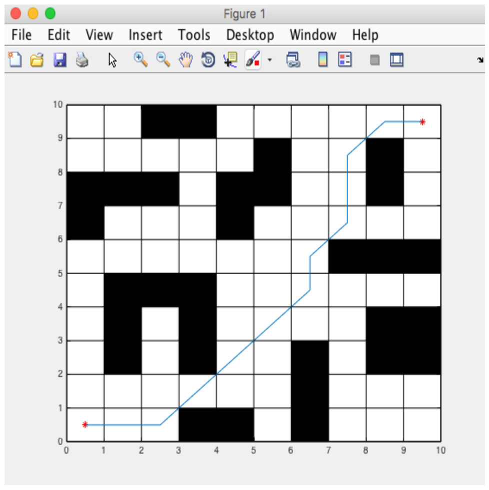
We calcuate the best performance index, time performance index and robustness performance index of GA-ACA and ACA-GA in map 02. The results display in Table 3.

In Table 3, we can clearly see that the best performance index of GA-ACA and ACA-GA are all equal to 0%, indicating that the ability of GA-ACA and ACA-GA to search the optimal solution is the same when dealing the grid environment maps contains one trap.

In terms of the timing performance index, ACA-GA is 0.70%, obviously lower than

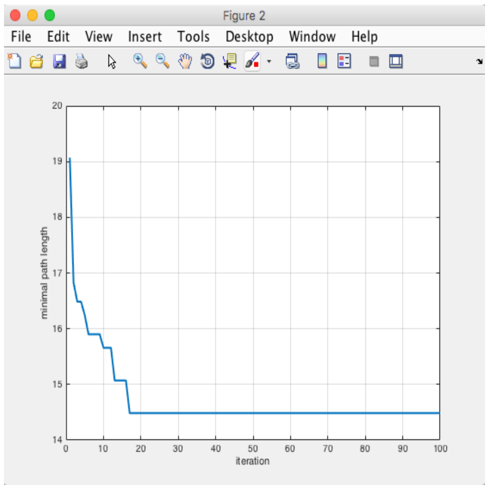


(a)

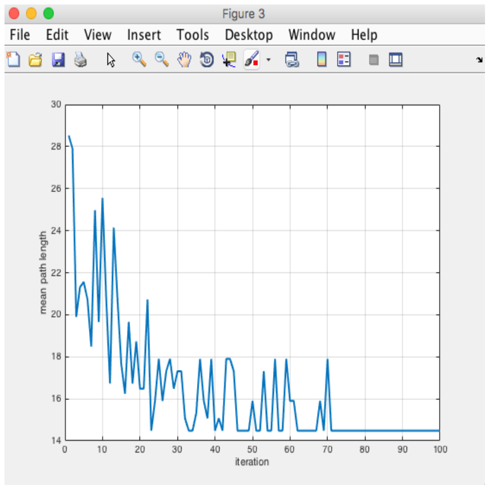


(b)

FIGURE 18: (a)The optimal path obtained by GA-ACA in map 02, the length of the optimal path is 14.4853; (b)The optimal path obtained by ACA-GA map 02, the length of the optimal path is 14.4853.



(a)



(b)

FIGURE 19: (a)The optimal path length evolution graph by using GA-ACA in map 02; (b)The mean path length evolution graph by using GA-ACA in map 02.

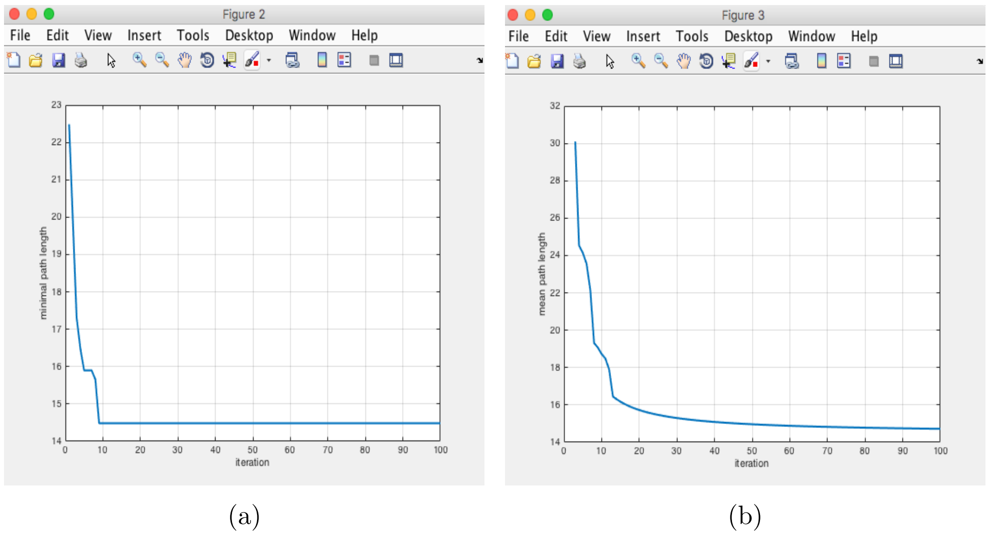


FIGURE 20: (a)The optimal path length evolution graph by using ACA-GA in map 02; (b)The mean path length evolution graph by using ACA-GA in map 02.

TABLE 3: The performance index of GA-ACA and ACA-GA in Experiment two

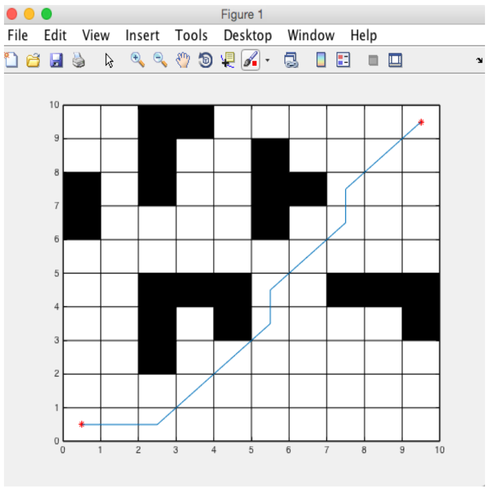
Algorithm name	Best (%)	Timing (%)	Robustness (%)	Iteration numbers	
				Optimal found	Optimal convergence
GA-ACA	0	3.60	3.64	17	71
ACA-GA	0	0.70	1.63	9	50

that of GA-ACA(3.60%), which demonstrates that the convergence rate of ACA-GA is higher than that of GA-ACA.

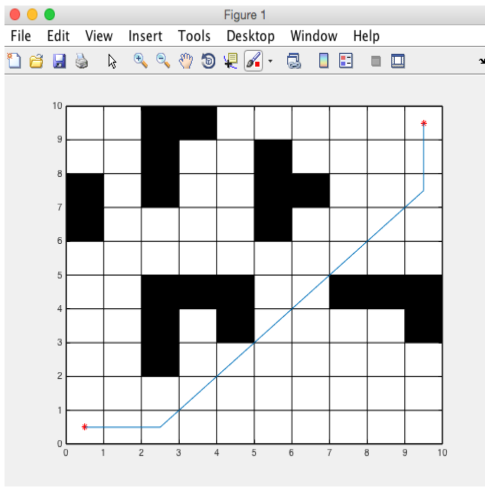
The last algorithm performance index is robustness, we can see that the robustness of ACA-GA(1.63%) is lower than that of GA-ACA(3.64%), which demonstrates that the stability of ACA-GA is much better than GA-ACA in the maps with one trap. In a word, ACA-GA performs better than GA-ACA in terms of algorithms' convergence speed and stability. It is more suitable to apply ACA-GA than GA-ACA when solving robot path planning problems in a grid map contains traps.

5.4.3 The Thrid Group of Experiments

In the second group of experiments, we can see that ACA-GA deal effectively with the grid environment maps with one deep trap, better than GA-ACA. If there are more deep traps in a grid environment map, what's the performance of these two



(a)



(b)

FIGURE 21: (a)The optimal path obtained by GA-ACA in map 03, the length of the optimal path is 13.8995; (b)The optimal path obtained by ACA-GA map 03, the length of the optimal path is 13.8995.

algorithms? We design two grid environment maps to compare GA-ACA and ACA-GA when these two algorithms solving path planning problems in the maps with different diversity. The map 03 (shown in Fig.21) has 20 obstacle grids with one deep trap, and the map 04 (shown in Fig.22) has 40 obstacle grids with two deep traps, which has greater complexity than map 03. The optimal path obtained by GA-ACA and ACA-GA in map 03 is shown in Fig.21. And The optimal path obtained by GA-ACA and ACA-GA in map 04 is shown in Fig.22.

We can see that the optimal path length obtained by GA-ACA and ACA-GA algorithm in map 03 is both 13.8995 from Fig.21, and the optimal path length obtained by GA-ACA and ACA-GA algorithm in map 04 is both 14.4853 from Fig.22. The results show that the search capability of GA-ACA and ACA-GA is the same no matter how the complicated the grid map is in the thrid group of experiment.

The results of the optimal path length evolution graph and the mean path length evolution graph are plotted by implementing GA-ACA (Fig.23), and ACA-GA (Fig.24) in map 03. In Fig.23, we can see that the optimal path is found at iteration 12, and the ant colony converge to the optimal path at iteration 60 by using GA-ACA; In Fig.24, the optimal path is found at iteration 10, and the ant colony converge to the

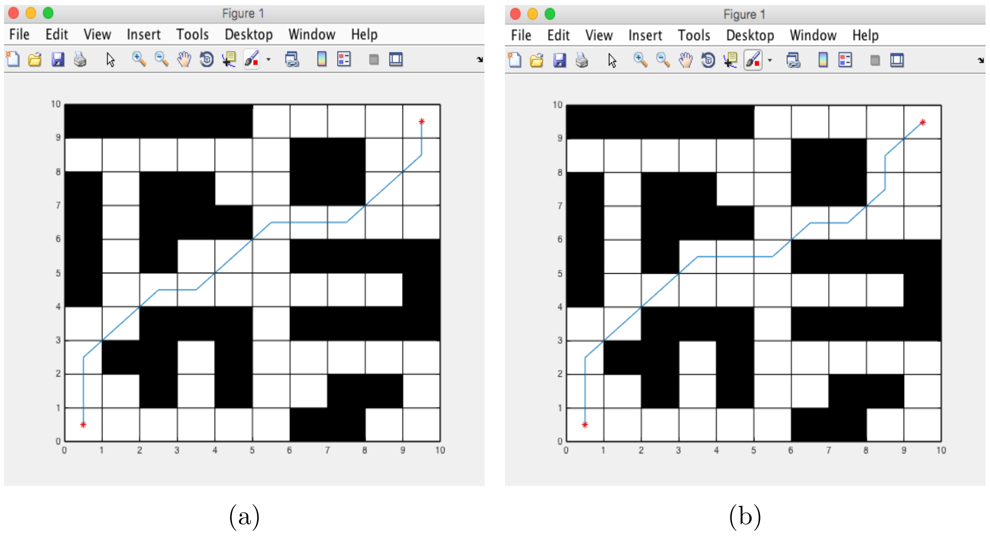


FIGURE 22: (a)The optimal path obtained by GA-ACA in map 04, the length of the optimal path is 14.4853; (b)The optimal path obtained by ACA-GA map 04, the length of the optimal path is 14.4853.

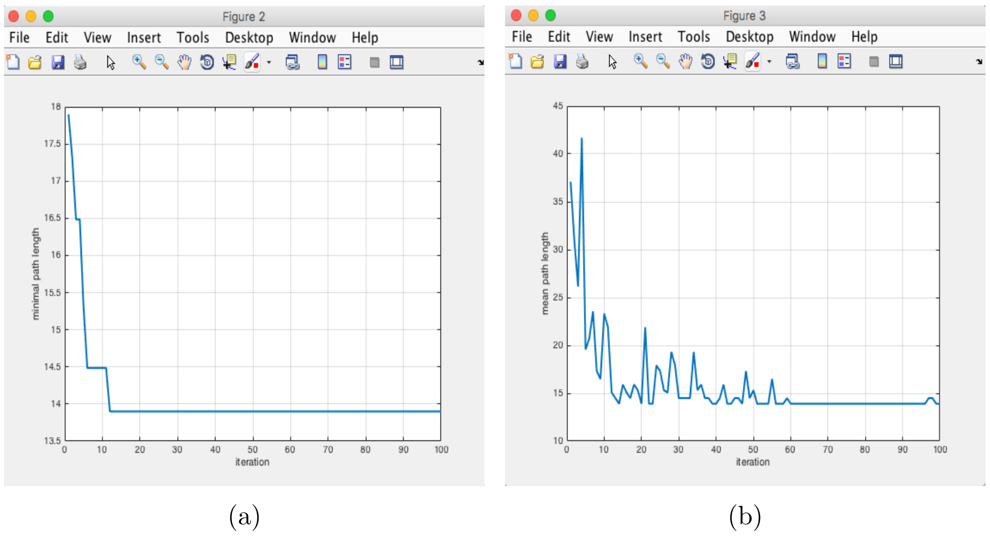
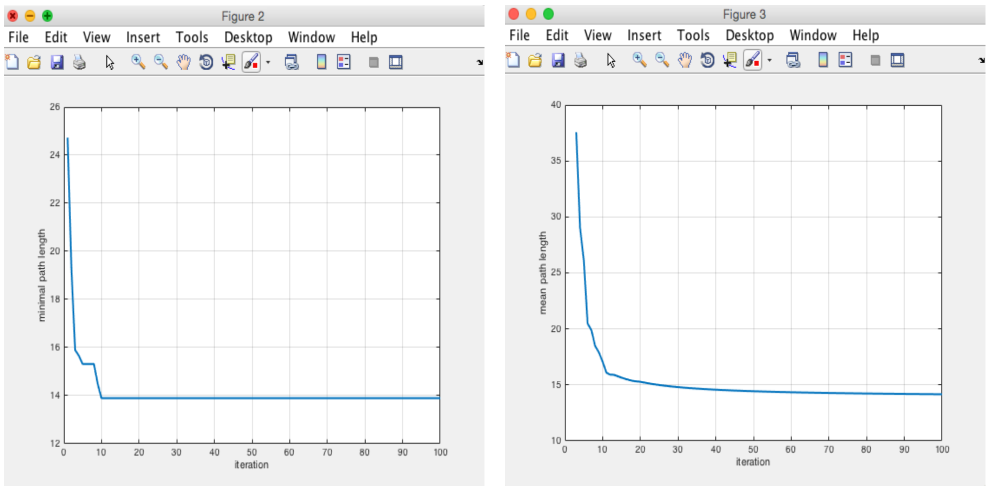
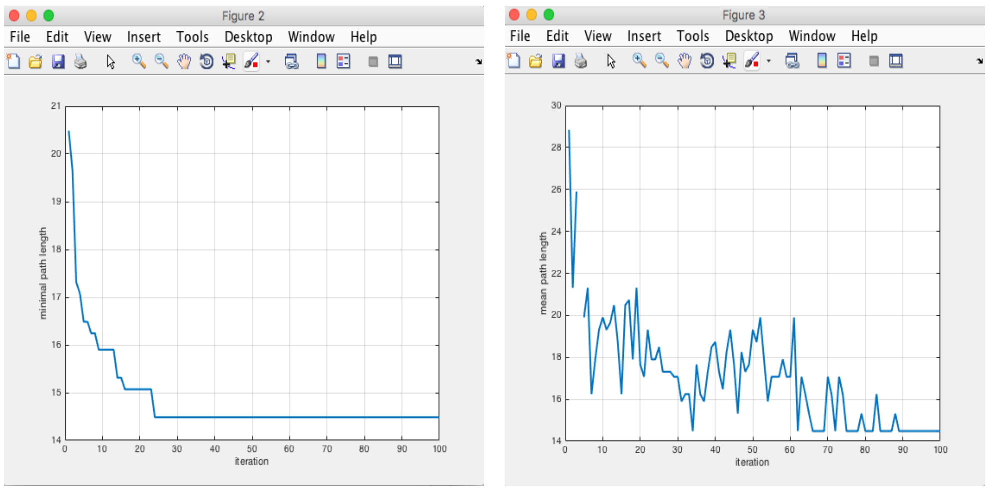


FIGURE 23: (a)The optimal path length evolution graph by using GA-ACA in map 03; (b)The mean path length evolution graph by using GA-ACA in map 03.



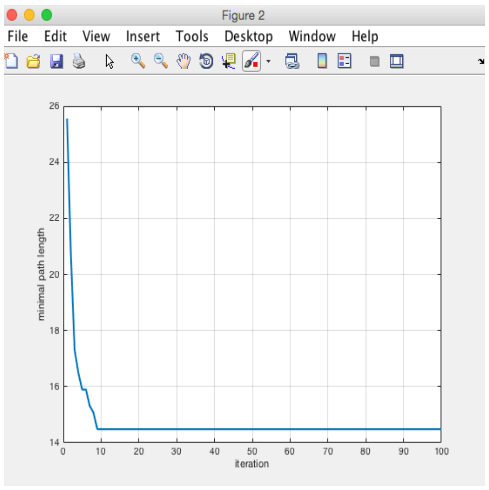
(a) (b)

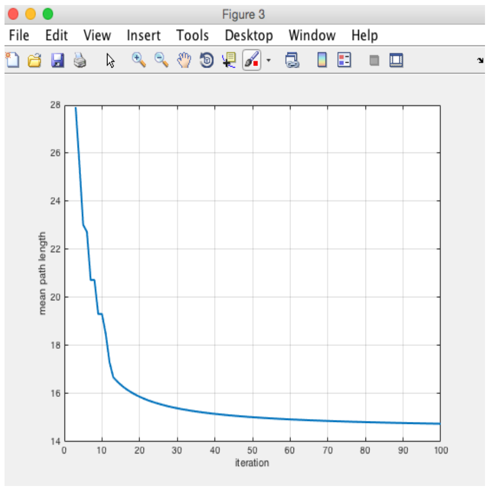
FIGURE 24: (a)The optimal path length evolution graph by using ACA-GA in map 03; (b)The mean path length evolution graph by using ACA-GA in map 03.



(a) (b)

FIGURE 25: (a)The optimal path length evolution graph by using GA-ACA in map 04; (b)The mean path length evolution graph by using GA-ACA in map 04.





(a) (b)

FIGURE 26: (a)The optimal path length evolution graph by using ACA-GA in map 04; (b)The mean path length evolution graph by using ACA-GA in map 04.

TABLE 4: The performance index of GA-ACA and ACA-GA in Experiment three under map 03

Algorithm name	Best (%)	Timing (%)	Robustness (%)	Iteration numbers	
				Optimal found	Optimal convergence
GA-ACA	0	1.62	5.99	12	60
ACA-GA	0	0.88	0.82	10	40

optimal path at iteration 40 by using ACA-GA; Similarly, we get the optimal path length evolution graph and the mean path length evolution graph by implementing GA-ACA (Fig.25) , and ACA-GA (Fig.26) in map 04. In Fig.25, we can see that the optimal path is found at iteration 24, and the ant colony converge to the optimal path at iteration 89 by using GA-ACA; In Fig.26, the optimal path is found at iteration 9, and the ant colony converge to the optimal path at iteration 50 by using ACA-GA; These results show that the more complicated of the grid environment map, the slower of the algorithm's convergence speed. Also, we can see that the convergence speed of ACA-GA is more quick than GA-ACA no matter the complexity of the maps.

Next, we calcuate the best performance index, time performance index and robustness performance index of GA-ACA and ACA-GA in map 03 and map 04. The results display in Table 4 and Table 5.

In Table 4, we can clearly see that the best performance index of GA-ACA and

TABLE 5: The performance index of GA-ACA and ACA-GA in Experiment three under map 04

Algorithm name	Best (%)	Timing (%)	Robustness (%)	Iteration numbers	
				Optimal found	Optimal convergence
GA-ACA	0	2.23	14.31	24	89
ACA-GA	0	0.82	1.82	9	50

ACA-GA are all equal to 0%. Likewise, the best performance index of GA-ACA and ACA-GA are all equal to 0% in Table 5.

In terms of the timing performance index, the Table 4 shows that ACA-GA is 0.88%, lower than that of GA-ACA(1.62%) in map 03. The Table 5 shows that the timing performance index of GA-ACA(2.23%) is also greater than that of ACA-GA(0.82%) in map 04.

The last algorithm performance index is robustness index, the Table 4 shows that ACA-GA is 0.82%, lower than that of GA-ACA(5.99%) in map 03. The Table 5 shows that the timing performance index of GA-ACA(14.31%) is obviously greater than that of ACA-GA(1.82%) in map 04.

These three algorithm performance indexes demonstrate that the ability of GA-ACA and ACA-GA to search the optimal solution is same and the convergence speed and stability of ACA-GA is higher than that of GA-ACA no matter how the complexity of the map is in my experiment. Another interesting thing we can see is that the more complicated of the map, the better of ACA-GA performs in terms of the algorithm's stability.

5.4.4 The Fourth Group of Experiments

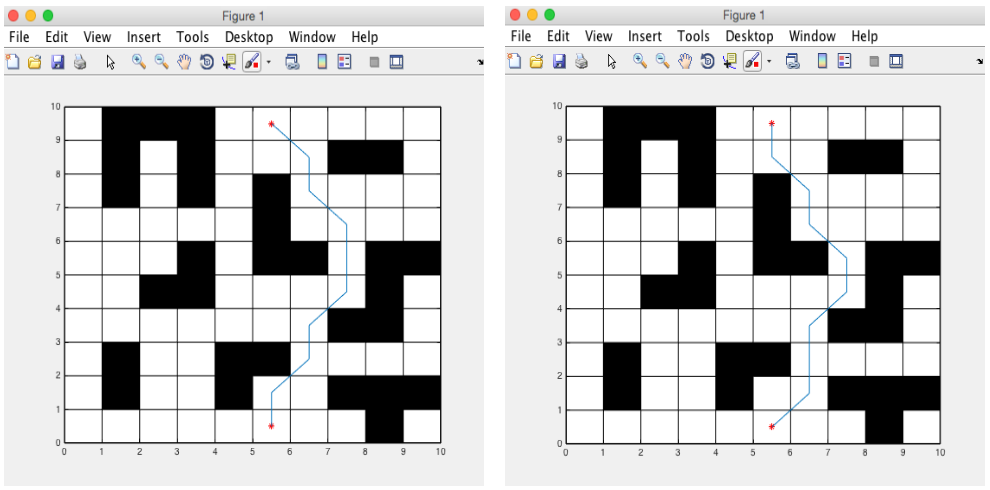
In the first three groups of experiments, the common condition is that the starting point is 1 and the destination point is 100. However, the starting point and the destination point are not fixed in the real robot path planning problem. Therefore, we set two different pairs of the starting point and the destination point to compare algorithms' performance by using GA-ACA and ACA-GA under the grid environment map 05 (shown in Fig.27 and Fig.28). Firstly, we set the starting point to grid 6 and the destination point to grid 96, the optimal path obtained by GA-ACA and ACA-GA is shown in Fig.27. Secondly, we switch the starting point and the destination point to 41 and 70, the optimal path obtained by GA-ACA and ACA-GA is shown in Fig.28.

We can see that the optimal path length obtained by GA-ACA and ACA-GA algorithm in map 05(the starting point is 6, the destination point is 96) is both 10.6569 from Fig.27, and the optimal path length obtained by GA-ACA and ACA-GA algorithm in map 05(the starting point is 41, the destination point is 70) is both 10.6569 from Fig.28. The results show that the search capability of GA-ACA and ACA-GA is alike no matter where the starting point and the destination point are in the forth group of experiment.

The results of the optimal path length evolution graph and the mean path length evolution graph are plotted by implementing GA-ACA (Fig.29), and ACA-GA (Fig.30) in map 05(6-96). In Fig.29(a), we can see that the optimal path is found at iteration 4, and ant colony converge to the optimal path at iteration 68 by using GA-ACA in Fig.29(b); In Fig.30(a), the optimal path is found at iteration 9, and ant colony converge to the optimal path at iteration 40 by using ACA-GA in Fig.30(b); Similarly, we obtain the optimal path length evolution graph and the mean path length evolution graph by implementing GA-ACA (Fig.31), and ACA-GA in (Fig.32) in map 05(41-70). In Fig.31(a), we can see that the optimal path is found at iteration 10, and ant colony converge to the optimal path at iteration 86 by using GA-ACA Fig.31(b); In Fig.32(a), the optimal path is found at iteration 11, and ant colony converge to the optimal path at iteration 50 by using ACA-GA in Fig.32(b);

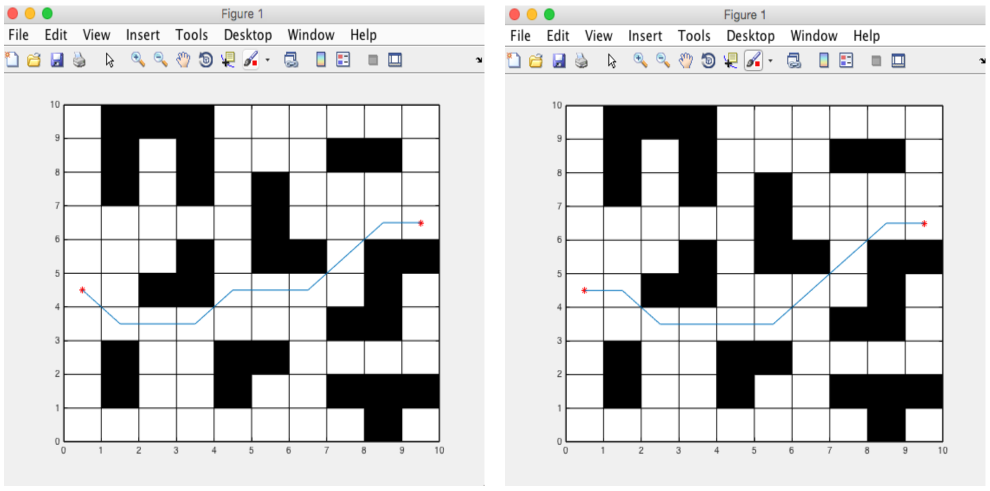
Next, we calcuate the best performance index, time performance index and robustness performance index of GA-ACA and ACA-GA under map 05. The results of map 05 display in Table 6.

In Table 6, we can clearly see that the best performance index of GA-ACA and ACA-GA are all equal to 0% when the algorithm's starting point is 6 and the destination point is 96. Likewise, the best performance index of GA-ACA and ACA-GA are all equal to 0% when the algorithm's starting point is 41 and the destination point is



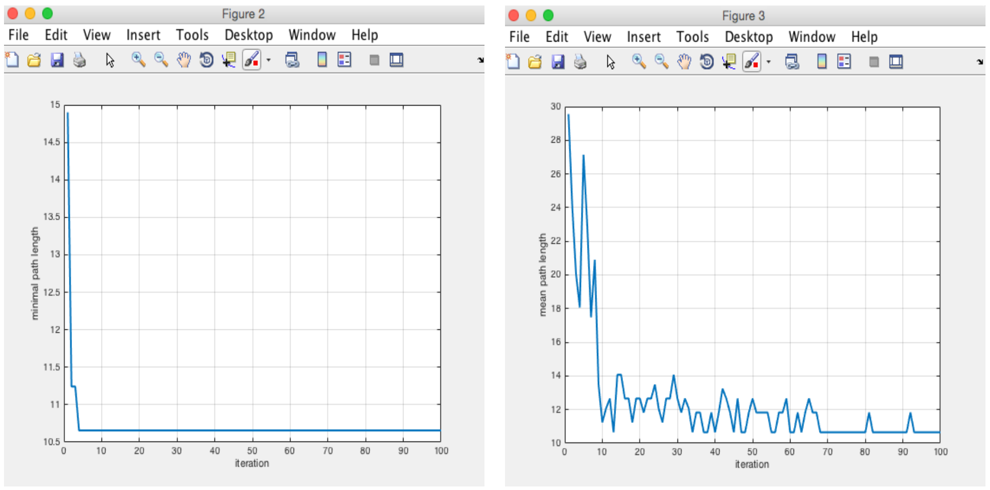
(a) (b)

FIGURE 27: (a)The optimal path obtained by GA-ACA in map 05(6-96), the length of the optimal path is 10.6569; (b)The optimal path obtained by ACA-GA map 05(6-96), the length of the optimal path is 10.6569.



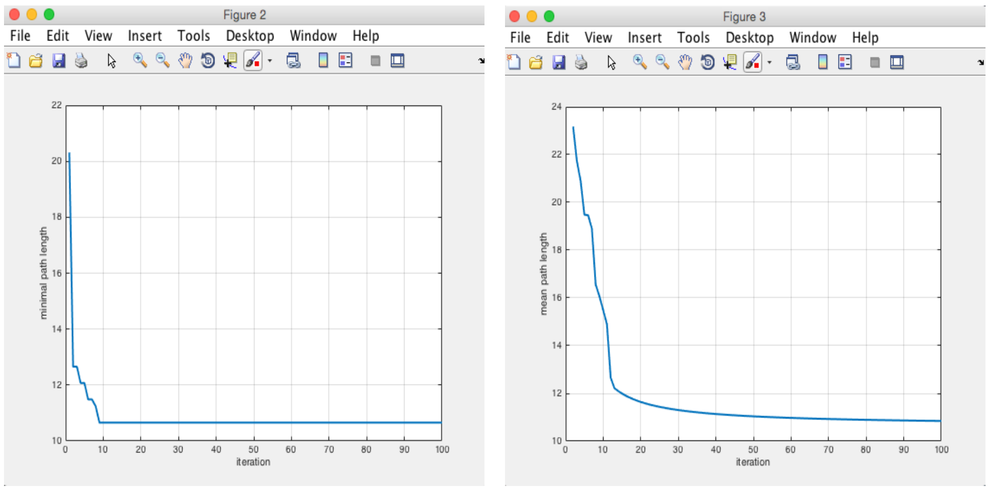
(a) (b)

FIGURE 28: (a)The optimal path obtained by GA-ACA in map 05(41-70), the length of the optimal path is 10.6569; (b)The optimal path obtained by ACA-GA map 05(41-70), the length of the optimal path is 10.6569.



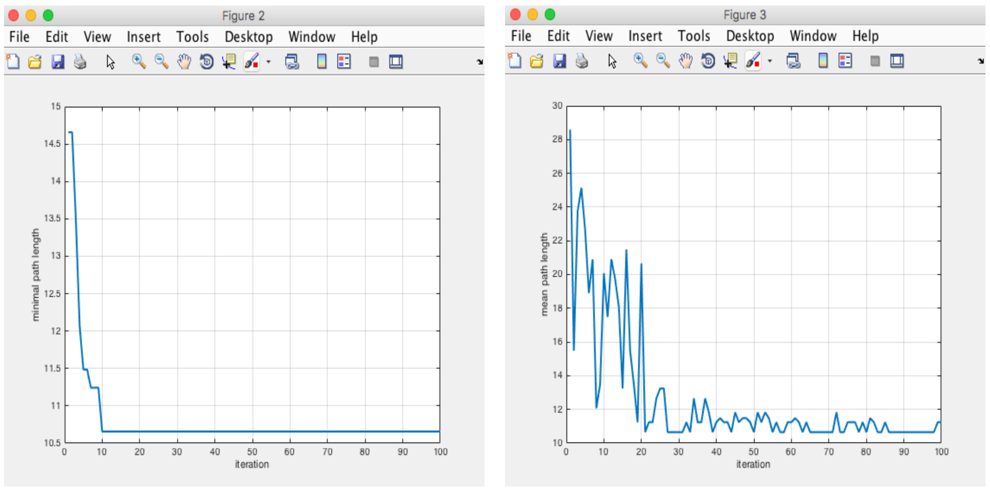
(a) (b)

FIGURE 29: (a)The optimal path length evolution graph by using GA-ACA in map 05(6-96); (b)The mean path length evolution graph by using GA-ACA in map 05(6-96).



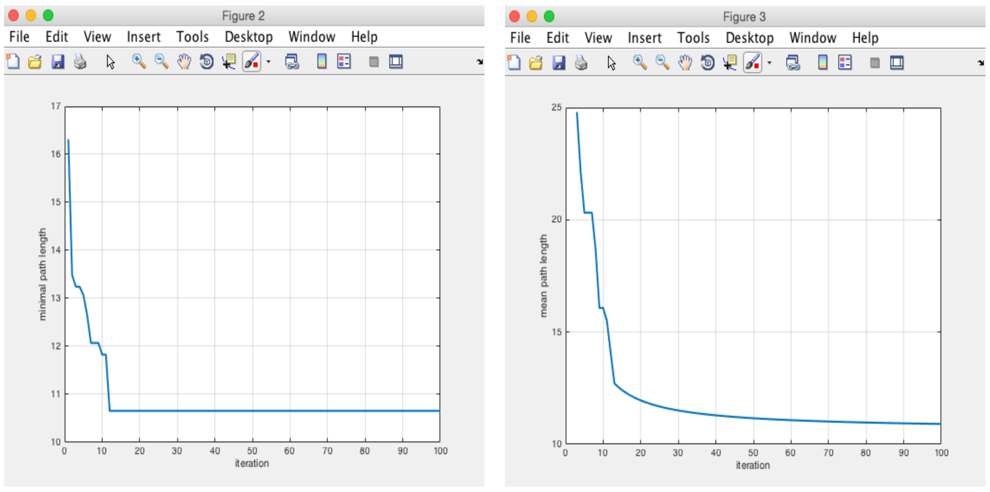
(a) (b)

FIGURE 30: (a)The optimal path length evolution graph by using ACA-GA in map 05(6-96); (b)The mean path length evolution graph by using ACA-GA in map 05(6-96).



(a) (b)

FIGURE 31: (a)The optimal path length evolution graph by using GA-ACA in map 05(41-70); (b)The mean path length evolution graph by using GA-ACA in map 05(41-70).



(a) (b)

FIGURE 32: (a)The optimal path length evolution graph by using ACA-GA in map 05(41-70); (b)The mean path length evolution graph by using ACA-GA in map 05(41-70).

TABLE 6: The performance index of GA-ACA and ACA-GA in Experiment four

S-D	Algorithm	Best (%)	Timing (%)	Robustness (%)	Iteration numbers	
					Optimal found	Optimal convergence
6-96	GA-ACA	0	0.97	7.11	4	68
	ACA-GA	0	0.93	1.76	9	40
41-70	GA-ACA	0	1.02	8.13	10	86
	ACA-GA	0	0.94	2.31	11	50

70.

In terms of the timing performance index, the Table 6 shows that the timing index of ACA-GA is 0.93%, little bit lower than that of GA-ACA(0.97%) when the algorithm's starting point is 6 and the destination point is 96. The results performs alike when the algorithm's starting point is 41 and the destination point is 70.

The last algorithm performance index is robustness, the Table 6 shows that ACA-GA is 1.76%, lower than that of GA-ACA(7.11%) when the algorithm's starting point is 6 and the destination point is 96. The timing performance index of GA-ACA(8.13%) is also obviously greater than that of ACA-GA(2.31%) when the algorithm's starting point is 41 and the destination point is 70.

These three algorithm performance indexes demonstrate that the ability of GA-ACA and ACA-GA to search the optimal solution is the same and the convergence speed and stability of ACA-GA is higher than that of GA-ACA no matter how the starting point and the destination point vary.

CHAPTER 6

Conclusion and Future Work

This thesis is mainly focus on the mobile robot path planning problems based on ant colony optimization algorithm combined with genetic algorithm, namely GA-ACA and ACA-GA. To be specific, the research works of the thesis are listed as below:

Firstly, the actual working environment of the mobile robot is modeled. Environmental modeling adopts the grid method. The actual working environment is divided into grids of the same size, and the grids containing the obstacles are grayed out, called obstacles grid; the other grids called free grid filled with white. We combine the method of serial number and rectangular coordinate to identify all grids, and these two methods can be converted to each other. After setting the starting grid and the target grid of the mobile robot path planning, the robot path planning based on the ant colony optimization algorithm is actually a process that through the interaction and mutual cooperation between the individual of the ant colony, they will avoid all obstacles grids to finally find an optimal path from the starting grid to the target grid.

Secondly, the thesis has made some modifications in order to accommodate ACA to robot path planning in grid-based environment. For example, the thesis apply pseudo-random-proportional rule instead of random-proportional rule to choose path and redifine the heuristic information η_{ij} from page 25-27 in chapter 3.2 .

At last, based on MATLAB platform, the thesis designs four comparative experiments to verify the validity and effectiveness of the GA-ACA and ACA-GA algorithm under different grid maps. Through the results of four groups of experiment, we can summarize several conclusions:

- The ability of GA-ACA and ACA-GA algorithm to search the optimal solution, convergence speed and stability are higher than that of the ant colony algorithm and that of genetic algorithm;
- When dealing the grid environment maps contains traps, the ability of GA-ACA and ACA-GA to search the optimal solution is same, however, ACA-GA performs better than GA-ACA in terms of algorithms' convergence speed and stability;
- The ability of GA-ACA and ACA-GA to search the optimal solution is same and the convergence speed and stability of ACA-GA is higher than that of GA-ACA no matter how the complexity of the map and how the starting point and the destination point vary.
- In conclusion, ACA-GA is more suitable and effective than GA-ACA in solving robot path planning problem considering algorithm's processing time and stability.

However, the research work in this thesis also has some limitations, mainly reflecting in the following aspects:

- This thesis only set the grid map size is 10*10. The future work is to do more experiments under multiple enlarged grid maps.
- This thesis studies the single robot path planning problem in a static environment, and the reality is that there is a need for multi-robotic systems. If the algorithms proposed in this paper be applied to the path planning problems in multi-robotic systems, we need to do more research work.
- This thesis deals with the problem of robot path planning in static environment with known environmental information. However, the robot in real workspace is likely to be in a dynamic environment where the environmental information is completely unknown or partially unknown. The limitation causes the proposed algorithms (GA-ACA; ACA-GA) not be directly applied to the robot path planning in dynamic environment.

REFERENCES

[1] Borenstein, J. and Koren, Y. (1991). The vector field histogram-fast obstacle avoidance for mobile robots. *IEEE Transactions on Robotics and Automation*, 7(3):278–288.

[2] Brooks, R. A. and Lozano-Perez, T. (1985). A subdivision algorithm in configuration space for findpath with rotation. *IEEE Transactions on Systems, Man, and Cybernetics*, (2):224–233.

[3] Canny, J. and Reif, J. (1987). New lower bound techniques for robot motion planning problems. In *Foundations of Computer Science, 1987., 28th Annual Symposium on*, pages 49–60. IEEE.

[4] Dorigo, M. (1992). Optimization, learning and natural algorithms. *Ph. D. Thesis, Politecnico di Milano, Italy.*

[5] Dorigo, M., Birattari, M., and Stutzle, T. (2006). Ant colony optimization. *IEEE computational intelligence magazine*, 1(4):28–39.

[6] Dorigo, M. and Blum, C. (2005). Ant colony optimization theory: A survey. *Theoretical computer science*, 344(2-3):243–278.

[7] Dorigo, M. and Gambardella, L. M. (1997). Ant colonies for the travelling salesman problem. *biosystems*, 43(2):73–81.

[8] Dorigo, M., Maniezzo, V., and Colorni, A. (1996). Ant system: optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 26(1):29–41.

[9] Dozier, G., Esterline, A., Homaifar, A., and Bikdash, M. (1997). Hybrid evolutionary motion planning via visibility-based repair. In *Evolutionary Computation, 1997., IEEE International Conference on*, pages 507–511. IEEE.

[10] Duan, H. B. (2005). *Ant Colony Algorithms: Theory and Applications*. Beijing Science Press.

[11] Fiorini, P. and Shiller, Z. (1998). Motion planning in dynamic environments using velocity obstacles. *The International Journal of Robotics Research*, 17(7):760–772.

[12] Fox, D., Burgard, W., and Thrun, S. (1997). The dynamic window approach to collision avoidance. *IEEE Robotics & Automation Magazine*, 4(1):23–33.

[13] Gambardella, L. M. and Dorigo, M. (1996). Solving symmetric and asymmetric tsps by ant colonies. In *Evolutionary Computation, 1996., Proceedings of IEEE International Conference on*, pages 622–627. IEEE.

[14] Guan-Zheng, T., Huan, H., and Sloman, A. (2007). Ant colony system algorithm for real-time globally optimal path planning of mobile robots. *Acta automatica sinica*, 33(3):279–285.

[15] HE, J., TU, Z.-y., and NIU, Y.-g. (2010). A method of mobile robotic path planning based on integrating of GA and ACO [j]. *Computer Simulation*, 3:045.

[16] Ishida, T. (2004). Development of a small biped entertainment robot QRIO. In *Micro-Nanomechatronics and Human Science, 2004 and The Fourth Symposium Micro-Nanomechatronics for Information-Based Society, 2004. Proceedings of the 2004 International Symposium on*, pages 23–28. IEEE.

[17] Kennedy, J. (2011). Particle swarm optimization. In *Encyclopedia of machine learning*, pages 760–766. Springer.

[18] Khatib, O. (1986). Real-time obstacle avoidance for manipulators and mobile robots. In *Autonomous robot vehicles*, pages 396–404. Springer.

[19] Lozano-Perez, T. (1990). Spatial planning: A configuration space approach. In *Autonomous robot vehicles*, pages 259–271. Springer.

[20] Lozano-Pérez, T. and Wesley, M. A. (1979). An algorithm for planning collision-free paths among polyhedral obstacles. *Communications of the ACM*, 22(10):560–570.

[21] Martınez-Alfaro, H. and Gómez-Garcıa, S. (1998). Mobile robot path planning and tracking using simulated annealing and fuzzy logic control. *Expert Systems with Applications*, 15(3):421–429.

[22] Mei, H., Tian, Y., and Zu, L. (2006). A hybrid ant colony optimization algorithm for path planning of robot in dynamic environment. *International Journal of Information Technology*, 12(3):78–88.

[23] Miao, H. and Tian, Y.-C. (2008). Robot path planning in dynamic environments using a simulated annealing based approach. In *Control, Automation, Robotics and Vision, 2008. ICARCV 2008. 10th International Conference on*, pages 1253–1258. IEEE.

[24] Mitchell, M. (1998). *An introduction to genetic algorithms.* MIT press.

[25] Nilsson, N. J. (1984). Shakey the robot. Technical report, DTIC Document.

[26] Raja, P. and Pugazhenthi, S. (2012). Optimal path planning of mobile robots: A review. *International Journal of Physical Sciences*, 7(9):1314–1320.

[27] Reimann, M., Doerner, K., and Hartl, R. F. (2004). D-ants: Savings based ants divide and conquer the vehicle routing problem. *Computers & Operations Research*, 31(4):563–591.

[28] Siegwart, R., Nourbakhsh, I. R., and Scaramuzza, D. (2011). *Introduction to autonomous mobile robots.* MIT press.

[29] Stützle, T. and Hoos, H. H. (2000). Max–min ant system. *Future generation computer systems*, 16(8):889–914.

[30] Vadakkepat, P., Tan, K. C., and Ming-Liang, W. (2000). Evolutionary artificial potential fields and their application in real time robot path planning. In *Evolutionary Computation, 2000. Proceedings of the 2000 Congress on*, volume 1, pages 256–263. IEEE.

[31] Veloso, M. M., Rybski, P. E., Lenser, S., Chernova, S., and Vail, D. (2006). Cmrobobits: Creating an intelligent AIBO robot. *AI magazine*, 27(1):67.

[32] Wang, Y. and Cai, Z. X. (2009). Intelligent optimization algorithm and its application. *Computer Education*, 11:034.

[33] Wang, Y., Mulvaney, D., and Sillitoe, I. (2006). Genetic-based mobile robot path planning using vertex heuristics. In *Cybernetics and Intelligent Systems, 2006 IEEE Conference on*, pages 1–6. IEEE.

[34] Whitley, D. (1994). A genetic algorithm tutorial. *Statistics and computing*, 4(2):65–85.

[35] Zhang, L. X., Wang, Y. X., Wang, B. B., Deng, Q., and Chen, H. (2012). Path planning for mobile robot based on ACA-GA. In *Applied Mechanics and Materials*, volume 135, pages 673–677. Trans Tech Publ.

VITA AUCTORIS

NAME:	Chenhan Wang
PLACE OF BIRTH:	Shenyang, Liaoning province, China
YEAR OF BIRTH:	1990
EDUCATION:	University of Science and Technology Liaoning, B.Eng., Computer Science and Technology, AnShan, China, 2013
	University of Windsor, M.Sc in Computer Science, Windsor, Ontario, 2017